Ipso Facto

The A-C-E Magazine

INDEX ISSUE #16 PAGE

MARCH 1980.

A.C.E. EXECUTIVE COMMITTEE.....2
EDITORIAL COMMENTS.....3
THE MONITOR.....4
A COMPARISON OF NETRONICS AND QUEST: "FULL BASIC".....19
ROLL OVER HEX WHILE LOADING.....21
SOFTWARE FOR THE ETI 2708 EPROM PROGRAMMER.....22
LETTERS TO THE EDITORS.....24
SOME THOUGHTS ON DEVICE INDEPENDENT I/O.....25
THE 1/Nth DEGREE.....27
8K EPROM MEMORY BOARD.....28
SOME THOUGHTS ON THE FUTURE OF THE 1802.....30
FUTURE MEETINGS AND NEW LOCATION.....32
ITEMS FOR SALE.....33
ASSEMBLY LISTING FOR COSMAC GRAPHICS PROGRAM.....33
PROTO BOARD.....40
BYTE CONTROL #84.....43
MINUTES OF ACE MEETINGS.....50
64K DECODING FOR TEKTRON MEMORY.....51
CUDDLY SOFTWARE: A PRODUCT REVIEW.....52
ERRATA.....54
PALO ALTO TINY BASIC INTERPRETER.....55
NEW PRODUCT: SUPER ISOLATOR.....55
MEMBERSHIP INFORMATION.....56

IPSO FACTO is published by the ASSOCIATION OF COMPUTER EXPERIMENTERS (A.C.E), a non-profit, educational organization. Information contained in IPSO FACTO is believed to be accurate and reliable. However, no responsibility is assumed by IPSO FACTO or the ASSOCIATION OF COMPUTER EXPERIMENTERS for its use; nor for any infringements of patents or other rights of third parties which may result from its use.

Send newsletter correspondence to Bernie Murphy,
102 McCraney St.,
Oakville, Ontario,
Canada. L6H 1H6

>

1979-1980 ACE EXECUTIVE COMMITTEE

PRESIDENT	Ken Bevis	220 Cherry Post Rd. Toronto, ONT., L5A 1H9 416-277-2495
TREASURER	Mike Franklin	24 Duby Rd, Acton, ONT., L7J 2P1 519-853-3421
ASSOCIATE EDITOR	Vic Kushnir	2640 Barnstone Cr., Mississauga, ONT., L5K 2C1 416-822-6505
ASSOCIATE EDITOR	Earle Laycock	2772 Hollington Cres., Mississauga, ONT, L5K 1E7 416-823-1345
ASSOCIATE EDITOR	Bob Silcox	562 Forestwood Cr., Burlington, ONT., L7L 4K3 416-681-2848
CONSULTING EDITOR	Bernie Murphy	102 McCraney St., Oakville, ONT., L6H 1H6 416-845-1630
DRAFTING	J. Myszkowski	99 Augusta St., Hamilton, ONT. 416-529-0250
MEMBERSHIP CO-ORDINATOR	Blair Gerrish	2110 Prospect St., Apt. 3, Burlington, Ont., L7R 1Y9 416-634-0268
HARDWARE CO-ORDINATOR	Fred Feaver	105 Townsend Ave., Burlington, ONT., L7T 1Y8 416-637-2513
PROGRAM CO-ORDINATOR	Fred Pluthero	1013 Upper Wellington St., Hamilton, ONT., L9A 3S4 416-389-4070
PUBLISHING COMMITTEE	Dennis Mildon	44 Wildewood Ave., Hamilton, ONT., L8T 1X3 416-385-0798
	John Hanson	955 Harvey Place, Burlington, Ont. L7T 3E9 416-637-1076

EDITORIAL COMMENTS

As mentioned elsewhere in this issue, A.C.E. meetings will be returning to the Stelco Auditorium in Hamilton, beginning in April. The Oakville location which we have been using for the past few meetings was excellent in almost every respect, but unfortunately, turnout at these meetings has been rather poor -consistently so, and so we felt that if we were getting fewer members out to meetings, then apparently the membership in this area generally prefer the old location. If there are any who have not been coming out because of the move to Oakville, we hope to see you again starting with the April 8 meeting (starting time 7:30 P.M.).

As noted on the back cover, we are now entirely sold out of back issues 1-12, so please do not send in any further orders for them. Work is currently in progress on our collection of the best material from these issues, and this collection, tentatively titled 'De Facto', should be available before too long. We cannot yet specify the exact price (this will depend on a number of factors, such as number of pages that we wind up with), or the exact date of availability, so please do not send in orders at this time. We hope to be able to supply this information before too long -- in issue 18 at the latest. 'De Facto' will be sold as loose leaf pages, with 3-hole punch, a cardboard cover, and plastic wrapping so that it will survive the rigors of travelling through the postal system. This format will, among other advantages, keep the cost at a minimum, and make it fairly straight-forward to supply supplementary volumes for issues beyond number 12 at a later date.

We have received two offers to serve as Education Co-ordinator, and hope to feature material from one or both of these people in upcoming issues.

The annual meeting is still a couple of months away, but it is not too soon to start thinking about candidates for next year's for next year's executive. In particular, the position(s) associated with editing of the club newsletter may prove very difficult to fill. The success of Ipso Facto, and its consequent growth from a small newsletter into a more substantial publication, has brought with it new problems, in that producing it can no longer be handled by one person as it was originally. Availability of word-processing equipment, or similar automatic equipment is almost a necessity for producing it at its present size -- hand-typing a manuscript of this size for each issue would keep a typist busy for a long time. A recent executive meeting addressed this problem without any definite solution, after considering a number of ideas, such as assigning responsibility for each issue to a different group of people. Ideas on how we can keep Ipso Facto going at its present level, in terms of both quantity and quality, would be much appreciated. Even better would be offers to take part in this undertaking. Please give it some thought.
>

Steve Nies
2510 Deas St.
Bossier City, LA

71111

THE MONITOR

As I did more and more machine language programming on my Elf, I realized that I needed a monitor with more than the usual memory/cassette routines. After looking over all the monitors available for the 1802, I decided that none of them would meet all of my needs. That's when I decided to design a custom monitor for my Elf.

I then sat down and tried to think of what commands I would need now and in the future. After much consideration, the following 10 commands were chosen:

1. Execute - X,P can be set to any value except 0 or 1.
2. Memory Examine/Load.
3. Register Examine/Load.
4. Breakpoint Insertion.
5. Fill memory with a byte.
6. Move a block of memory from one location to another.
7. Search memory for a varible length string of bytes.
8. Cassette Routines -
   Save - Load - Verify at either 300 or 1200 baud using named files of any length up to 16 characters (modified Ipso Facto standard).

Another feature of this monitor is the ability to add to the command set without having to change any code in ROM. This is accomplished by having the table lookup routine vectored through System RAM. The first application of this feature will be the addition of an EDITOR - ASSEMBLER to the monitor. Also, future plans include a BASIC Pseudo-Compiler to be called by a monitor command. The actual operation of this feature will be discussed in more detail later on in the article.

Other miscellaneous niceities of this monitor include:

1. Totally transparent to the calling program. D,DF, and all registers except R(0) and R(1) are saved.
2. I/O vectored through System RAM. This allows the user to dynamically change the I/O device while the program is executing.
3. Eight Level Interrupts supported.
4. 12 utility subroutines available to the caller's program through the SCRT subroutine convention.

As mentioned earlier in the article, this monitor requires a block of memory to be dedicated as System RAM. This block (256 bytes) can be located anywhere in memory. However, I recommend 9800 since the Quest Super Elf already has memory there. The purpose of the System RAM is for temporary storage of the buffer, register save area, stack, and other varibles used by the monitor. Future additions to the monitor will also use this RAM. A memory map of the System RAM is shown in Figure 1.

After this monitor was written and debugged, I noticed I had a little memory left over in the P page. Not wanting to waste good memory, I decided to include software to drive a keyboard and memory mapped video display (since I already had the board). After talking to some friends, they asked if it would be possible to add a driver routine for a RS-232 device. Since the monitor was designed to be independent of I/O, there are no restrictions on the type of I/O software used. Even though I have not given much thought to an RS-232 scheme, I am sure the software Uart routines in the monitor could be used to accomplish this.

S 00 - S 3F	Input Buffer (64 characters wide)
S 40 - S 5B	Register Storage (R(2).1 at S 40, R(2).0 at S 41, ...R(F).0 at S 5B)
S 5C	DF of user's program stored
S 5D	D register stored
S 5E	X,P of user's program stored. This value defaults to 2,3. For other values of X,P , change this location to the desired value.
S 5F - S 63	Break location and contents stored
S 64 - S 66	Long jump to Input routine (defaults to P 97)
S 67 - S 69	Long jump to Output routine (defaults to P 9C)
S 6A - S 6C	Long jump to SoftWare Interrupt routine. When the user's program executes a SWI (bytes 22 79 D1), control is vectored through this location. (defaults to monitor location I 04)
S 6D - S 6F	End-of-table pointer. This pointer defaults to the area immediately after the monitor. It is in this area that I plan to put future additions to the monitor. If the user wishes to add other commands, this pointer can be changed to point to his table elsewhere in memory.
S 70 - S 7F	Interrupt Vector storage. Vector 7 is stored first, followed by vector 6, and so on. All vectors except vector 0 default to NOP. Vector 0 defaults to the SWI routine, location L 71.
S 80 - S 8F	Cassette file name storage. (Note: the file name is stored backwards)
S 90 - S CF	Unused
S D0 - S FF	Monitor stack.

Figure 1.

While on the subject of I/O, I should mention that all input to the monitor is first stored in a buffer. This allows the user to backspace over any errors and correct the mistakes before typing a CR. The only exception to this is in the Memory or Register Examine commands. While in this mode, any input from the I/O device is used directly by the command.

Before continuing on to discuss the commands, two bits of information concerning my notation might be helpful. First, the monitor does not require the commands to be entered in any particular format. The monitor will ignore any excess characters entered. As long as the information is in there somewhere, the monitor will accept it. For example, suppose the operator types 'Memory Examine Location 0300 Hex', the monitor will interpret this as 'ME 0300'. There are a few basic syntax rules, however.

1. All commands must begin in column 1.
2. There must be at least one space between the command and any parameters.
3. All parameters must consist of either 2 or 4 consecutive hex digits, depending on the parameter.

Note that with this arrangement, commands can be abbreviated. All that is needed by the monitor is the abbreviation that would make the command unique. Later on, in the discussion of the commands, the abbreviation of the command will be shown as both capitalized and underlined letters, with optional letters being lowercase (i.e. BReak).

Concerning the second part of my notation, all references to the high order byte of an address will be given by a underlined capital letter (with I being the base page). This was done to enable the user to easily relocate the monitor to any 2K block of memory. Any future references to the monitor (or any subroutines) should also use this notation. This way, everyone can maintain software compatibility while having the flexibility of locating the monitor anywhere.

Also, any future expansion of the monitor will use this notation, beginning with the letter Q. One exception to this convention was that the letter S was chosen to represent the page where the System RAM resides. As an example to this notation, I debugged this program this monitor beginning at 0800, with my System RAM at 9800. Therefore, I would equal 08, J equals 09, and so forth. Also, S would equal 98.

One last thing should be mentioned. While discussing the commands, the first parameter will be symbolically represented as AAAA, with the second parameter (as required) being BBBB, and so on. If a parameter only needs two digits, it will be shown as AA (for the first parameter), BB (for the second parameter) and so forth. Also, a CR will be shown as ↵.

With this in mind, the commands will each be listed separatly, with a short description on each.

1a. Execute AAAA↵

This command will transfer control to the user's program beginning at AAAA. Before transferring control to the program, D,DF, and all registers are restored. In order to use a different X and/or P value for the program, change the byte at location S 5E to the desired value. The new value of X,P is what the computer will use for the user's program.

1b. Execute↵

This command has two subfunctions. If X,P have not been changed, (X,P defaulted to 2,3) execution of the user's program will begin at location 0000. If X,P has been changed, execution will be transferred to the location pointed to by the register defined by P. This second function is useful after a breakpoint has been serviced. To resume execution after a break, simply type E↵. This will transfer control back to the point of the break with the original value of X,P that the user's program established.

Before control is passed to the user's program, the monitor will print 'SST?' and wait for an entry. This will allow the user to activate his single step circuitry. Note that the single step operation is accomplished by hardware only! If single step circuitry is not implemented on the system, change the byte at location L 33 from 39 to 30.

If the user does have single step capability, he can chose to single step through the program by pressing a 'Y' on the keyboard. The program then loops until the hardware is activated. Once the circuitry has been activated, pressing any key on the keyboard will start the single step operation.

If the user does not want to single step through the program, any key on the keyboard except 'Y' can be pressed. Control will then be passed to the target program.

2. MEmory examine AAAA↵

This command is probably the most used command in the monitor. It's purpose is to examine and/or load data into memory. After the command is entered, the monitor will print the location, a hyphen, and then the data. If the byte is also a valid ASCII character (between 20 and 7F), the monitor will also print the character. This feature is useful for observing text in memory.

There are five subcommands available. They are the carriage return, line feed, space, back space, and escape keys. There functions are as follows:

The carriage return key will close the current location and then open up the next location for examination/modification.

The line feed key will close the current location, increment the pointer by 8, and then print the contents of the new location.

Pressing the space bar will step through memory one nibble at a time. Any bytes spaced over are not changed.

Pressing the back space key will accomplish one of two things: 1) If the second nibble of the byte is being observed, the back space key will move the cursor back to the first nibble. 2) If the first nibble of the byte is being observed, pressing the back space key will close the current location, decrement the memory pointer, and then open up the new location.

Finally, the escape key will exit the Memory Examine command and reenter the monitor.

3. Register examine↵

This command is used to examine or modify the registers that were saved from the user's program. After the command is entered, the monitor will print the number of the register being observed (R(2) is the first register examined),a hyphen, and then the two-byte contents of the register. These registers are later restored to the original program through the use of the Execute command.

There are four options associated with this command. They are the carriage return, space, back space, and escape keys. These options are basically the same as the Memory Examine options, but with a few differences. The first difference is that the options now operate on two-byte data, instead of one byte. The second difference is that the monitor will not allow the user to space beyond R(F), or back space past R(2). If the user attempts this, the console bell will be rung.

4. SAve AA BB 'CCC...CC'↵

This command is used to save memory blocks (in increments of 256 bytes) on a storage device. The general form for this command is SA AA BB 'CCC...CC'↵ , where AA is the number of the starting page, BB is the number of the ending page, and 'CCC...CC' is the file name. The file name can be any length up to 16 characters. If a name is not 16 characters long, it will be padded on the right with blanks. If the name is longer than 16 characters, it will be truncated.

Before going on to the next command, some discussion on the format is needed. When I first designed these routines, I wanted it to be compatable with the Ipso Facto standard. However, I also wanted longer file names than one character. After thinking over the possibilities, I decided on the following format.

1) At least five seconds of mark (2400 hz) at the beginning of a file.

2) All data bytes are 8 bits, plus one start bit, even parity, and two stop bits.

3) Tape format -

file name	16 bytes	Any ASCII characters between 20 and 7F.
start address	2 bytes	(hi then lo byte)
byte count	2 bytes	(hi then lo byte)
data bytes n x 256	bytes	
checksum	1 byte	Checksum formed by adding all bytes excluding file name and checksum.

Besides the obvious advantage of having longer file names, another advantage of this format is the ability to have the data rate at either 300 or 1200 baud. In order to Save or Load files at 300 baud, the file name should begin with an 'S' as the first character of the name. If the file does not have the preceeding 'S', the monitor will assume a 1200 baud rate.

Finally, the monitor does not reguire any additional hardware. All Kansas City tones are software generated. Also, a routine has been included (the PLL routine) to receive and decode these tones into a binary 1 or 0.

The only other difference is the checksum byte. In my software, I did not complement the checksum, mainly because for a long time I did not know how this method worked. However, it took about the same number of bytes to do it either way, and I really did not want to hand re-assemble the monitor all over again! Since the standard was modified anyway, I decided that this difference was insignificent.

5. Load 'CCC...CC'↵

This command is used to load memory with a file name of 'CCC...CC' from a storage device. The monitor will search for this file name, while printing 'FOUND CCC...CC' for every other file it skips over. If the programmer wishes to only load the first file found, he can type L ''↵ .

Also, there is no need to select the baud rate. The monitor will select the baud rate from the first letter of the file name. Please refer to the Save command for an explanation of this format.

There are two possible error indications associated with this command. If a parity error occurs, the Q light will come on but loading will continue. If a checksum error occurs, the monitor will then print 'CHECKSUM?'. One exception to these conditions is if a parity error occurs in loading the header information. If an error occurs here, further loading will stop and control will be returned to the monitor. This feature was included to preserve memory from being wiped out by a misread tape.

6. Verify 'CCC...CC'↵

This command is the same as Load in every way except that instead of a file being loaded into memory, the file on the storage device is compared to the file already in memory. If the two files match, control will be passed back to the monitor. If the two files differ, the monitor will print 'ERROR IN XXXX'. One application for this command is for checking if a file was transmitted to tape successfully.

7. BReak AAAA↵

This command is used to set a breakpoint at AAAA. Before a breakpoint is inserted, the bytes at this and the next two locations are saved along with the address of the break. A SoftWare Interrupt Pseudo-op (bytes 22 79 D1) is then inserted. When the SWI is executed by the user's program, D, DF, and all registers except R(0) and R(1) are saved and control passes to the monitor. The monitor will then print 'BREAK AT AAAA' and then restores the bytes that were saved earlier. Note that due to this process, only one breakpoint may be inserted at a time. However, immediately after a break has been serviced, another breakpoint may be inserted.

One word of caution, for the break routine to work properly, either a Priority Interrupt I.C. must be connected to port 6, or the port must not be used. If this port is being used, change the bytes at locations L 4E and L 67 to a free port. If the routine still does not work, make sure that when the interrupt port is read, an FF is returned.

One disadvantage to this command is that a breakpoint can only be inserted in RAM. However, in later extensions of the monitor, I plan to add interrupt driven breakpoint and SST software to eliminate this problem.

After a break has been serviced, execution of the user's program can continue by using the Execute command. Please refer to the section on this command for further information.

8. MOve AAAA BBBB CCCC↵

The purpose of this command is to move the contents of the memory from AAAA to BBBB to the new location beginning at CCCC.

9. Fill AAAA BBBB CC↵

This command will fill memory from AAAA to BBBB with the byte CC.

10. SEarch AAAA BBBB 'CC CC .... CC'↵

This command is used to search memory for a string of bytes. The general form for this command is SE AAAA BBBB 'CC CC .... CC', where AAAA is the search starting address, BBBB is the search ending address,

and 'CC CC .... CC' is the byte string to be searched for. Note that the string can be any length up to 25 bytes long. After this command is entered, the monitor will begin searching the target area. If the string is found, the monitor will print 'FOUND AT XXXX' for every occurance of the string. After the monitor prints a location, it will then wait for any key on the keyboard to be pressed. The user should then press a key to start the search again. After all searching is finished, a prompt will be printed to indicate to the user that a new command may be entered. If the monitor does not find the string, it will print 'NOT FOUND' and return control back to the monitor.

One very useful application of this command is in converting programs from one 1802 system to run on another system. I have already used this command to convert Netronic's monitor to run on my system and in relocating Quest's monitor to lower memory (by finding and changing all long branches).

Besides these 10 commands, it is possible to add more commands to the monitor. This feature is due to the way the monitor handles unrecognized commands.

After searching the table contained in the monitor, the COMM-REC routine will then run across a 01 byte. This byte signals the routine that the next two bytes contain the address of the next table to be scanned. This address in the monitor ROM points to the location S 6D. This location in System RAM was initilized to 01 Q 00. After the search is transferred to S 6D, the COMM-REC routine hits another 01 byte. This time the new location is Q 00 (the first location past the monitor). After transferring the search to Q 00 (where there should not be any memory), the monitor will start the search again. Since the byte at this location should be greater than 60 hex, the monitor will realize that the command table has been exhausted. Finally the monitor will print 'COMMAND?'.

In order to temporarily add commands to the monitor, the user should change locations S 6E and S 6F to point to his table of new commands. This table should consist of the new commands in the following format:

Command Abbreviation	Delimiter Byte	Address of routine
XXXX....	00	XXXX

Following the table of new commands, a FF byte should be inserted. This will signal the COMM-REC routine that no more tables will follow.

Note that if the user adds some temporary commands, he must change the end-of-table pointer (in System RAM) every time he enters the monitor. If he wishes to add permanent commands, all that is required is to locate the new command table at location Q 00. There will be no need to change any other locations.

As mentioned elsewhere in the article, this monitor was designed to be system independent. However, there are a few locations in the listing that should be changed to reflect differences in the 1802 systems. These locations are shown below:

Location	Byte	Purpose
I 00	62	Turn off 1861 Video I.C.
L 35	3D	Loop on no keyboard input flag
L 4E	6E	Read interrupt vector port
L 67	66	Write to interrupt acknowledge port
P 82	36	Cassette input flag
P 88	3E	Cassette input flag

This monitor has been extensively tested for any errors on several systems. So far, the monitor has been successfully used on the Quest Super Elf and Netronics Elf II. However, one quirk was discovered while running the monitor on a friend's system. This quirk concerns the way the output handles a carriage return.

When I was designing this monitor, I was using an output device that simply used a CR (byte 0D) to start a new line. Therefore, I designed the print routines based on this fact. However, the terminal on my friend's system needs a carriage return and a line feed. This is not a major problem. All that is needed is to change the output vector to a free space in memory. In this space the user needs to insert a little routine that tests the output characters for a CR. When a CR is detected, a CR LF should then be output. Note that if the user has a memory mapped display, he may use the video driver already in the monitor without worrying about this problem.

MONITOR LISTING

```
I 00  62 C4                        Turn off 1861
  02  7A 38 7B
  05  A1                           Save Accumulator
  06  F8 I B3 F8 0D A3 D3          Set R(3) as PC
  0D  E3 71 03                     Disable Interrupts
  10  F8 S B0 F8 5D A0             Save D,DF in System RAM
  16  81 73 7E FA 01 73
  1C  93 B1 F8 B0 A1 D1            Call REGSAVE
  22  90 B9 B2 F8 J B5 B4          Initilize Registers
  29  F8 16 A4                     .
  2C  F8 26 A5
  2F  F8 FF A2
  32  D4 I FD                     Call RAM_INIT
  35  S 64                           RAM Load Address
  37  C0 P 97 C0 P 9C                System I/O Vectors
  3D  C0 I 04                        SWI Program Vector
  40  01 Q 00                        End-of-Table Ptr.
  43  L EC L EC L EC L EC            Interrupt Vectors 7 - 4
  4B  L EC L EC L EC L 71            Interrupt Vectors 3 - 0
  53  FF                             Subroutine Delimiter
  54  78 02 AD                     Store T Register
  57  F8 5E A7                     Save X,P Default Value
  5A  F8 23 57
  5D  D4 J 11                      Clear Screen
  60  0C 00
  62  39 98                        Do BREAK Routine if Q Set
  64  D4 J 11                      Print 'BREAK AT '
  67  42 52 45 41 4B
  6C  20 41 54 20 00
  71  F8 5E A9                     Save T Register
  74  8D 59 19
  77  49 BA 49 AA                  Restore Saved Bytes
  7B  1A 1A EA
  7E  49 73 49 73 09 9A
  84  8D FA 0F FE FC 3C A9        Save Location of Break
  8B  9A 59 19 8A 59
  90  D4 K 95                     Print Location
  93  D4 J 11                      Print CR
  96  0D 00
  98  D4 J 11                      Print Prompt
  9B  3E 00
  9D  F8 40 A9                     Fill Buffer with Blanks
  A0  29 F8 20 59
  A4  89 3A A0
  A7  D4 J 36                      Call INBUFF
  AA  D4 J 65                      Call COMM-REC
  AD  30 98
```

```
                        /* REGSAVE */
I AF  D3                                    Return
  B0  8F 73 9F 73 8E 73 9E 73               Save Registers
  B8  8D 73 9D 73 8C 73 9C 73
  C0  8B 73 9B 73 8A 73 9A 73
  C8  89 73 99 73 88 73 98 73
  D0  87 73 97 73 86 73 96 73
  D8  85 73 95 73 84 73 94 73
  E0  F8 00 73 73 82 73 92 73
  E8  30 AF

                        /* BELL */
  EA  F8 20 A7 F8 20                        Initilize Duration, Pitch
  EF  31 F3 7B 38                           Alternate On/Off
  F3  7A
  F4  FF 01 3A F4                           Do Pitch Delay
  F8  27 87 3A ED                           Do Duration Delay
  FC  D5

                        /* RAM-INIT */
  FD  46 B7 46 A7                           Get Load Address
J 01  06 FB FF 32 0B                        Test for Delimiter
  06  46 57 17                             Store Data Byte
  09  30 01
  0B  16 D5                                 Return

                        /* PSTRING */
  0D  AA                                    Save Character
  0E  D4 S 67                               Print Character
  11  46 3A 0D                              Test for Delimiter
  14  D5                                    Return

                        /* CALL */
  15  D3                                    Return
  16  E2 96 73 86 73                        Save R(6)
  1B  93 B6 83 A6                           Save PC in R(6)
  1F  46 B3 46 A3                           Load Sub Address into PC
  23  30 15

                        /* RETURN */
  25  D3                                    Return
  26  96 B3 86 A3                           Load Return Address into PC
  2A  E2 12 72 A6 F0 B6                     Restore R(6)
  30  30 25
```

```
                        /* INBUFF */
J 32  89 32 56                              Test for Start of Buffer
  35  29
  36  D4 S 64                               Get Character
  39  D4 S 67                               Print Character
  3C  8A FB 0D 3A 42                        Is it a CR?
  41  D5                                    Yes, so Return
  42  8A FB 08 32 32                        Is it a Backspace?
  47  8A FB 07 32 56                        Is it a Bell?
  4C  89 FF 3F 3B 5B                        Ignore Char. if Buffer Full
  51  D4 J 11                               Print Backspace
  54  08 00
  56  D4 I EA                               Ring Bell
  59  30 36
  5B  8A FF 20 3B 36                        Valid Character?
  60  8A 59 19 30 36                        Yes, so Store it in Buffer

                        /* COMM-REC */
  65  93 B7 F8 B7 A7                        Initilize Command Table Ptr.
  6A  F8 00 A9 E9                           Initilize Buffer Ptr.
  6E  07 FB 01 3A 7E                        Test for any other Tables
  73  17 47 BA 47 AA                        Yes, so Load New Ptr.
  78  9A B7 8A A7
  7C  30 6A
  7E  07 FF 60 3B 92                        End of Table?
  83  D4 J 11                               Yes, so Print 'COMMAND?'
  86  43 4F 4D 4D 41
  8B  4E 44 3F 0D 00
  90  D5                                    Return
  91  19
  92  47 F3 32 9D                           Test for Match
  96  47 3A 96 17 17                        Inc. Ptr. to next Command
  9B  30 6A
  9D  07 3A 91                              Test for end of Command
  A0  89 FF 40 33 83                        Test for Delimiter
  A5  49 FB 20 3A A0
  AA  93 BA F8 B0 AA DA                     Set PC to R(A)
  B0  17 47 B3 47 A3 E2 D3                  Load Command Address and Exec.

  B7  45 00 K BD                            (Execute)
  BB  4D 45 00 M 75                         (Memory Examine)
  C0  52 00 N 00                            (Register Examine)
  C4  53 41 00 N B5                         (Save)
  C9  4C 00 Ø 80                            (Load)
  CD  56 00 Ø 83                            (Verify)
  D1  42 52 00 L CC                         (Break)
  D6  4D 4F 00 L AB                         (Move)
  DB  46 00 L 89                            (Fill)
  DF  53 45 00 L ED                         (Search)
  E4  01 S 6D                               End of Table Ptr.
```

```
                        /* H - A */
J E7  8A F6 F6 F6 F6 BA             Split Byte into two
   ED FF 0A 9A 3B F9                     Hex Characters
   F2 FF 09 F9 40 BA                Convert High Hex Nibble
   F7 30 FC                              to ASCII
   F9 F9 30 BA
   FC 8A FA 0F AA                   Convert Low Hex Nibble
K 00 FF 0A 8A 3B 0B                      to ASCII
   05 FF 09 F9 40 AA
   0A D5                            Return
   0B F9 30 AA
   0E D5                            Return

                        /* A - H */
   0F 8A FF 30 3B 2B                Test for Valid Character
   14 FF 0A 3B 24
   18 FF 07 3B 2B
   1C FF 06 33 2B
   20 8A FC 09 AA
   24 8A FA 0F AA                   Return Hex Nibble
   28 FF 00 D5                      Set DF, Return
   2B FC 00 D5                      Reset DF, Return

                        /* TEST */
   2E 8E 52 8F F7                   Compare Addresses
   32 9E 52 9F 77                   DF Set if R(F) ≥ R(E)
   36 D5

                    /* EXPR, INBYTE */
   37 F8 04 C8                      Expr Entry
   3A F8 02 B7                      Inbyte Entry
   3D 97 A7
   3F 89 FF 40 33 2B                Test for No Parameters
   44 49 AA
   46 D4 K 0F                       Call A - H
   49 3B 3D                         Invalid Character?
   4B 27 87 F6 3B 58
   50 8A FE FE FE FE AD             Shift Nibble to High
   56 30 3F                               Order Half of Byte
   58 8A 52 8D F1 AD                Merge Nibbles to get Byte
   5D 87 32 28                      Set DF if Finished
   60 8D BD 30 3F
```

```
                    /* CHAR-STORE */
   64 D4 K 0F                       Call A - H
   67 33 79                         Valid Character?
   69 D4 J 11                       No, so Print 'PARAMETER?'
   6C 50 41 52 41 4D 45
   72 54 45 52 3F 0D 00
   78 D5                            Return
   79 ED
   7A 8B F6 33 8B
   7E 0D FA 0F 5D
   82 8A FE FE FE FE F1 5D          Place Nibble in High
   89 1B D5                               Order Half of Byte
   8B 0D FA F0 5D
   8F 8A F1 5D                      Place Nibble in Low
   92 1B 1D                               Order Half of Byte
   94 D5                            Return

                        /* PWORD */
   95 8A 73 9A AA                   Convert First Two Bytes
   99 D4 J E7                             to ASCII
   9C 8A 73 9A AA                   Print First Nibble
   A0 D4 S 67
   A3 12 02 AA                      Print Second Nibble
   A6 D4 S 67
   A9 12 02 AA                      Continue through PCHAR

                        /* PCHAR */
   AC D4 J E7                       Convert Next Two Bytes
   AF 8A 73 9A AA                         to ASCII
   B3 D4 S 67                       Print First Nibble
   B6 12 02 AA                      Print Second Nibble
   B9 D4 S 67
   BC D5                            Return

                        /* EXEC */
   BD 09 FB 20 32 D6                Test for No Parameters
   C2 D4 K 37                       Call EXPR
   C5 3B 69                         Valid Character?
   C7 F8 5E A9                      Get X,P
   CA 09 FA 0F FE FC 3C A9          Store Address in Correct
   D1 9D 59 19 8D 59                      Register
   D6 D4 J 11                       Print 'SST?'
   D9 53 53 54 3F 00
   DE D4 S 64                       Get Answer
   E1 D4 S 67                       Print Answer
   E4 8A FB 59 32 EB                If Yes, Set Q
   E9 7A 38 7B
   EC 92 B0 F8 40 A0 E0
   F2 93 B1 F8 F8 A1 D1             Jump to REG-LOAD
```

```
                        /* REGLOAD */

F8  72 B2 72 A2 72 B3 72 A3     Restore Registers
00  72 B4 72 A4 72 B5 72 A5
08  72 B6 72 A6 72 B7 72 A7
10  72 B8 72 A8 72 B9 72 A9
18  72 BA 72 AA 72 BB 72 AB
20  72 BC 72 AC 72 BD 72 AD
28  72 BE 72 AE 72 BF 72 AF
30  72 F6 72                    Continue through Interrupt

                        /* INTERRUPT */

33  39 37                       Skip SST if Q Reset
35  3D 35                       Wait for Operator
37  70 38                       Return to Interrupted Program
39  D3                          Go Execute Subroutine
3A  E1 71 21 22 78 22 73 7E 73  Save T, D and DF
43  93 73 83 73 96 73 86 73     Save R(3) and R(6)
4B  F8 S B6 6E FA 07 FE FC 70 A6  Get Interrupt Vector
55  46 B3 46 A3
59  91 B6 F8 60 A6              Set Up Return Pointer
5E  30 39
60  93 B1 F8 66 A1 D1           Set PC to R(1)
66  52 66
68  42 A3 42 B3                 Restore R(3)
6C  42 F6 42                    Restore DF and D
6F  30 37

                        /* SW1 */

71  93 B0 F8 77 A0 D0           Set PC to R(0)
77  12 42 A6 42 B6              Restore R(6)
7C  42 A3 42 B3                 Restore R(3)
80  42 F6 42                    Restore DF and D
83  12 12 12                    Restore R(2)
86  C0 S 6A                     Jump to SWI Program Vector

                        /* FILL */

89  D4 K 37                     Get "From" Parameter
8C  CB K 69                     Invalid Character?
8F  9D BE 8D AE                 No, so Save in R(E)
93  D4 K 37                     Get "To" Parameter
96  3B 8C
98  9D BF 8D AF                 Save in R(F)
9C  D4 K 3A                     Get Filler Byte
9F  3B 8C
A1  D4 K 2E                     Finished?
A4  3B EC                       Yes, so Return
A6  8D 5E 1E 30 A1              No, so Store Byte

                        /* MOVE */

L AB  D4 K 37                   Get "From" Parameter
  AE  3B 8C                     Invalid Character?
  B0  9D BE 8D AE               No, so Save in R(E)
  B4  D4 K 37                   Get "To" Parameter
  B7  3B 8C
  B9  9D BF 8D AF               Save in R(F)
  BD  D4 K 37                   Get New Location
  C0  3B 8C
  C2  D4 K 2E                   Finished Moving?
  C5  3B EC                     Yes, so Return
  C7  4E 5D 1D 30 C2

                        /* BREAK */

  CC  D4 K 37                   Get Location
  CF  3B 8C                     Invalid Character?
  D1  92 B7 F8 63 A7 E7
  D7  4D 73 4D 73 0D 73         Save Old Contents of Location
  DD  F8 D1 5D 2D F8 79 5D 2D   Insert SWI Pseudo-Op
  E5  F8 22 5D
  E8  8D 73 9D 73               Save Location
  EC  D5

                        /* SEARCH */

  ED  D4 K 37                   Get "From" Parameter
  F0  3B 8C
  F2  9D BE 8D AE               Save in R(E)
  F6  D4 K 37                   Get "To" Parameter
  F9  3B 8C
  FB  9D BF 8D AF               Save in R(F)
  FF  8B FA FC AB               Reset Flags
M 03  89 FF 40 33 16            Search for Open Quote
  08  49 AA FB 27 3A 03
  0E  D4 K 2E                   Test for End of Search Range
  11  3B 5D
  13  D4 K 3A                   Get Byte to be Searched For
  16  CB K 69
  19  EE 8D F3 32 2B            Test for Match
  1E  29 09 FB 27 3A 1E         No Match, so Back Up Pointer
  24  19                              to Open Quote
  25  8B FA FE AB               Reset "Found" Flag
  29  30 34
  2B  8B F6 33 34
  2F  1B                        Set "Found" Flag
  30  8E AC 9E BC               Save Location of String
  34  1E                        Increment Search Pointer
  35  09 FB 27 3A 0E            Continue Search if No Close Quote
```

```
M 3A  8B F9 02 AB              Set "Found one" Flag
  3E  D4 J 11                  Print 'FOUND AT'
  41  46 4F 55 4E 44
  46  20 41 54 20 00
  4B  9C BA 8C AA
  4F  D4 K 95                  Print Location
  52  D4 J 11                  Print CR
  55  0D 00
  57  D4 S 64                  Wait until a key is pressed
  5A  2E 30 1E
  5D  8B FA 02 3A 6F
  62  D4 J 11                  Print 'NOT FOUND'
  65  4E 4F 54 20 46
  6A  4F 55 4E 44 00
  6F  D4 J 11                  Print CR
  72  0D 00
  74  D5                       Return

                    /* MEXAM */

  75  D4 K 37                  Get Location
  78  3B 16
  7A  9D BA 8D AA
  7E  D4 K 95                  Print Location
  81  D4 J 11                  Print ' - '
  84  20 2D 20 00
  88  0D AA                   Get Byte at Current Location
  8A  D4 K AC
  8D  D4 J 11                  Print Space
  90  20 00
  92  0D AA FF 20 3B 9C        If Byte is a valid ASCII
  98  FF 60 3B 9F                   Character, Print It
  9C  F8 20 AA
  9F  D4 S 67
  A2  D4 J 11                  Print Space
  A5  20 00
  A7  8B FA F8 AB              Reset Flags
  AB  D4 S 64                  Get new data or operation
  AE  8A FB 0D 32 D9                Test for CR?
  B3  FB 05 32 DC                   Backspace?
  B7  FB 02 32 EC                   Line Feed?
  BB  FB 2A 32 F6                   Space?
  BF  FB 3B 32 6F                   Escape?
  C3  8A FF 20 3B AB           Check for valid character?
  C8  D4 S 67                  Print it
  CB  D4 K 64                  Store it
  CE  8B F6 33 AB              Test for complete byte
  D2  D4 J 11                  Print CR
  D5  0D 00
  D7  30 7A
  D9  1D 30 D2                 Increment Pointer
  DC  8B F6 3B E9              Test for First Nibble
  E0  2B                       No, so backspace
  E1  F8 08 AA                 Print Backspace
  E4  D4 S 67
  E7  30 AB

N E9  2D 30 D2                 Decrement Pointer
  EC  8D FC 08 AD              Increment Pointer by 8
  F0  9D 7C 00 BD
  F4  30 D2
  F6  8B F6 33 D9              Test for First Nibble
  FA  1B                       Yes, So Space
  FB  F8 20 AA                 Print Space
  FE  30 E4

                    /* REXAM */

N 00  92 BD F8 40 AD           Set Up Register Pointer
  05  8D F6 FF 1E AA           Calculate Number of Register
  0A  D4 J E7                  Convert Number to ASCII
  0D  D4 S 67                  Print It
  10  D4 J 11                  Print ' - '
  13  20 2D 20 00
  17  4D BA 0D AA 2D           Print Register Contents
  1C  D4 K 95
  1F  D4 J 11                  Print Space
  22  20 00
  24  8B FA F8 AB              Reset Flags
  28  D4 S 64                  Get New Data or Operation
  2B  8A FB 1B C2 M 6F         Test for ESC?
  31  8A FB 0D 32 66                CR?
  36  8A FB 08 32 73                Backspace?
  3B  8A FB 20 32 96                Space?
  40  8A FF 20 3B 28           Test for Valid Character
  45  8D FF 5C 3B 4F           Test for Last Register
  4A  D4 I EA                  Yes, So Ring Bell
  4D  30 28
  4F  D4 S 67                  Print Character
  52  D4 K 64                  Store It
  55  8B FA 03 3A 28           Test for First Nibble
  5A  D4 J 11                  Print CR
  5D  0D 00
  5F  8D FF 5C 3B 05           Test for Last Register
  64  30 28                    Yes, So get Operation Only
  66  8D FF 5A 33 4A           Test for Last Register
  6B  1D                       No, So Increment Pointer
  6C  8D F6 3B 5A                   to Next Full Register
  70  1D 30 5A                      Location
  73  8D FF 42 33 7D           Test for First Register
  78  8B FA 03 32 4A           Yes, So Ring Bell if First Nibble
  7D  8B F6 3B 8A
  81  2B
  82  F8 08 AA                 Print Backspace
  85  D4 S 67
  88  30 28
  8A  8B FA 02 32 92           Decrement Pointer to Next Full
  8F  2D 30 81                      Register Word
  92  2D 2D 30 5A
```

```
N 96  8D FF 5A 3B A2          Test for Last Register
  9B  8B FA 03 FF 03 32 4A    Ring Bell if Last Nibble
  A2  8B F6 33 AC
  A6  1B
  A7  F8 20 AA                Print Space
  AA  30 85
  AC  1B 1D
  AE  8B FA 03 32 5A
  B3  30 A7

                    /* SAVE */

  B5  D4 K 3A                 Get "From" Block
  B8  CB K 69
  BB  8D BE
  BD  D4 K 3A                 Get "To" Block
  C0  3B B8
  C2  9E 52 8D F7 33 C9       Calculate Length
  C8  D5                      Return if Negative Length
  C9  FC 01 BD
  CC  89 FF 3F 33 B8          Test for end of Buffer
  D1  49 FB 27 3A CC          Search for Open Quote
  D6  FF 00
  D8  F8 10 BA BC             Init. Name length, Cycle Ctr.
  DC  F8 Ø B7 F8 57 A7        Init. OSC Subroutine
  E2  93 BF F8 EA AF DF       Set PC to R(F)
  E8  30 F7
  EA  F8 06 B0                Init. Header Delay
  ED  F8 00 AD AE
  F1  D7                      Call OSC
  F2  20 90 3A ED             Decrement Header Delay
  F6  D3                      Set PC to R(3)
  F7  97 BF F8 2D AF          Init. TX UART Subroutine
  FC  09 FB 53 32 05          Test for File Name of 'S...
Ø 01  9C F6 F6 BC             Divide #Cycle by 4
  05  09 FB 27 3A 0D          End of File Name?
  0A  F8 20 38                Yes, so pad with blanks
  0D  49 DF                   Transmit File Name
  0F  9A FF 01 BA 3A 05       Finished File Name?
  15  F8 00 BB                Zero Checksum
  18  9E DF 8E DF             Transmit Starting Address
  1C  9D DF 8D DF             Transmit Length
  20  4E DF                   Transmit Data
  22  2D                      Decrement Length
  23  9D 3A 20                Finished?
  26  8D 3A 20
  29  9B DF                   Transmit Checksum
  2B  D5                      Return
```



```
                  /* TX UART */

Ø 2C  D3                      Return
  2D  AA 73                   Save Character to be Transmitted.
  2F  F8 10 F6 AB AC          Init. DF, Parity, Bit Ctr.
  34  30 3A
  36  8A 76 AA 2C            Get Bit to be Transmitted
  3A  D7                      Call OSC
  3B  8C 3A 36                Finished?
  3E  8B F6                   Get Parity
  40  E2 E2                   Nop
  42  D7                      Transmit Parity
  43  12 9B F4 BB 22          Add Byte to Checksum
  48  FF 00                   Set DF
  4A  D7                      Transmit Stop Bit
  4B  C4 C4 E2 E2             Nop
  4F  FF 00                   Set Stop Bit
  51  D7                      Transmit Second Stop Bit
  52  12 02 30 2C             Restore Character

                   /* OSC */

  56  DF                      Return
  57  33 61                   Is the Bit a "1"
  59  9C F6 A1                Divide Cycle Ctr by 2
  5C  F8 0D 52                Init. Period Delay
  5F  30 68
  61  9C A1 1B
  64  F8 01 52 E2
  68  FF 01 3A 68             Delay one Period
  6C  21                      Decrement Cycle Ctr
  6D  39 72                   Alternate Q On/Off
  6F  7A 30 74
  72  7B 7B
  74  81 32 56                Return if Finished all Cycles
  77  F8 06                   Delay to balance half cycles
  79  FF 01 3A 79
  7D  02 30 68                Restore Period Delay

                /* LOAD, VERIFY */

  80  F8 00 C8                Load Entry Point
  83  F8 FF BC                Verify Entry Point
  86  E2
  87  7A                      Reset Parity Error Indicator
  88  92 BD F8 8F AD          Init. File Name Storage Ptr.
  8D  F8 P BF B7
  91  F8 7F A7                Init. PLL Subroutine
  94  F8 56 AF                Init. UART Subroutine
  97  F8 10 AE                Init. Name Length Ctr.
  9A  F8 00 A9                Reset Buffer
  9D  89 FF 3F C3 K 69        Test for End of Buffer
  A3  49 FB 27 3A 9D          Search for Open Quote
  A8  22 22                   Decrement Stack
  AA  09 FB 53 32 B2          Test for File of 'S....
  AF  F8 02 C8                Set Baud Rate for 1200 Baud
```

```
Ø B2  F8 08 52                  Set Baud Rate for 300 Baud
  B5  12 12                     Restore Stack
  B7  F8 02 BA                  Init. Header Length Delay
  BA  22 79 D7                  Call PLL
  BD  3B B7                     Test for Header
  BF  2A 9A 3A BA
  C3  DF 5D 2D 2E               Get File Name
  C7  09 FB 27 32 D0            Test for Close Quote
  CC  49 F3 3A D5               Test for Match
  D0  8E 3A C3
  D3  30 FD
  D5  DF 5D 2D 2E               No Match, So Save File Name
  D9  8E 3A D5
  DC  F8 8F AD F8 10 AE         Init. File Ptr., File Name Length
  E2  D4 J 11                   Print 'FOUND '
  E5  46 4F 55 4E 44 20 00
  EC  0D 2D AA                  Print File Name
  EF  D4 S 67
  F2  2E 8E 3A EC
  F6  D4 J 11                   Print CR
  F9  0D 00
  FB  30 87                     Go get Next File
  FD  F8 00 BB                  Clear Checksum
P 00  DF BE DF AE               Get Starting Address
  04  DF BD DF AD               Get Length
  08  C1 K 69                   Abort Load if Parity Error
  0B  9C 3A 2E                  Check Command Flag
  0E  DF 5B 1E 2D               Load Data
  12  9D 3A 0E                  Finished?
  15  8D 3A 0E
  18  DF 52 9B F7 F7 32 2D      Compare Checksums
  1F  D4 J 11
  22  43 48 45 43 4B           print 'CHECKSUM?'
  27  53 55 4D 3F 0D 00
  2D  D5                        Return
  2E  DF 52 0E F3 3A 3E         Verify Data
  34  2D 1E
  36  9D 3A 2E                  Finished?
  39  8D 3A 2E
  3C  30 18                     Go Compare Checksums
  3E  D4 J 11                   Print 'ERROR IN '
  41  45 52 52 4F 52
  46  20 49 4E 20 00
  4B  9E BA 8E AA
  4F  D4 K 95                   Print Address
  52  C0 M 6F                   Return
```

/* RX UART */

```
P 55  D3                        Return
  56  F8 08 AB AC               Init. Parity Ctr., Bit Ctr.
  5A  22 79 D7                  Wait for Stop Bit
  5D  33 5A
  5F  30 68
  61  22 79 D7                  Get Bit
  64  3B 67 1B                  Increment Parity if Bit = 1
  67  2C                        Decrement Bit Ctr.
  68  02 76 73                  Shift Bit into Byte
  6B  8C 3A 62                  Finished Byte?
  6E  79 D7                     Get Parity
  70  3B 73 1B
  73  9B F4 BB                  Add Byte to Checksum
  76  8B F6
  78  02 3B 55                  Return Byte
  7B  7B 30 55                  Set Q if Parity Error
```

/* PLL */

```
  7E  71                        Return
  7F  E2 42 A1                  Init. Cycle Ctr.
  82  36 82                     Wait for Low Cycle
  84  F8 90                     Init. Period Delay Constant
  86  FF 01 3E 86               Dec. Period while flag is low
  8A  FE                        Get Resultant Bit
  8B  33 91
  8D  21                        Decrement Cycle Ctr.
  8E  81 32 7E                  Return if done all cycles
  91  21                        Decrement Cycle Ctr.
  92  81 3A 82                  Repeat if all Cycles not Done
  95  30 7E
```

```
                                /* INPUT */

P 97  3D 97                          Wait for entry
  99  6D AA D5                       Read keyboard port

                                /* OUTPUT */

  9C  8A FF 20 3B C9                 Test for ASCII character
  A1  8A 58 18                       Print character
  A4  98 FF E2 3B C5                 Test for end-of-screen
  A9  F8 E0 B7 B8 F8 20 A7           Yes, so scroll up one line
  B0  47 58 18
  B3  97 FF E2 3B B0
  B8  F8 20 58 18                    Blank bottom line
  BC  98 FF E2 3B B8
  C1  28 F8 E0 A8
  C5  F8 2E 58 D5                    Print cursor and return
  C9  FB ED 32 F4                    Test for CR? (0D)
  CD  FB 01 32 E0                    Test for Form Feed? (0C)
  D1  FB 04 3A C8                    Test for Backspace? (08)
  D5  98 FF E1 33 DD                 Yes, so test for beginning-of-screen
  DA  88 32 C8                       Ignore backspace if cursor is at home
  DD  28 30 C5                       Backspace and print cursor
  E0  F8 E1 B8 F8 FF A8              Clear the screen
  E6  F8 20 58 28
  EA  98 FF E0 33 E6
  EF  E3 65 __                       Reset the Video Display I.C. (see note)
  F2  30 A3
  F4  88 FA E0 FC 20 A8              Start a new line
  FA  98 7C 00 B8
  FE  30 A4
```

Note: This I/O package is dependent on both the type of memory-mapped video display and the computer system itself. This package was designed to drive a video display based on the S68047 (or MC6847). If the display is not based on this I.C., change the three bytes at P EF to E2 E2 E2. These locations, if left unchanged, will output a byte to the S68047 to reset it to the alphanumeric mode. Since both the port and the byte itself is dependent on how the user has the I.C. wired, the control byte was left blank in the listing.

This I/O package assumes the video display is mapped at location E000. If the video board is at another location, change locations P A6, P AA, P B5, P BE, P D7, P E1, and P EC to reflect this difference.

The only three locations that are system dependent are P 97, P 99, and P F0. These locations are used to drive the input and output ports of a Quest Super Elf. Owners of different systems should change these locations to match their requirements.

Mike Franklin
24 Duby Road
Acton, Ontario
L7J 2P1

A Comparison of Netronics and Quest: "Full Basic"

GENERAL	NETRONICS	QUEST
Program address	0000-17FF	0000-29FF
Work space	last RAM page	2A00-2BFF
User space	1800--	2C00--
I/O	RS232 - out Q in EF4	RS232 - out Q in EF3
Variables	- unlimited - any combination of characters	- 26 - A-Z
ARRAYS	- none	- 26 A(exp) - Z(exp) max DIM 255
STRINGS	- unlimited - any combination of characters terminated with $ - 72 characters max	- 26 A$ - Z$ - 95 characters max
MATHAMATICS - format	- Reverse Polish Notation - Scientific notation ± 8 digit - Floating point ± 8 digit mantissa and 2 digit exponential value -	- normal - - Floating part ± 0.17 E38 5 digit mantissa - Interger ± 2147483648
Line # range	- 0 to 65535	- 1 to 65534, 65535 is auto "END"
line length	- 72 characters	- 95 characters
Execution mode	- direct - program (with line #)	- direct - program (with line #)
Serial Baud rate	to 300	to +3000

BASIC COMMAND SET

- identical or equivalent commands in both languages

REM	
CLS	- clean screen
CL#	- clear data
NEW	- clear program area
RUN	
PRINT or PR	
END	
LET	
GO TO	
IF ... THEN	

INPUT	- input data form keyboard
LIST	
GOSUB-RETURN	
FOR-STEP-NEXT	
PEEK	
POKE	
BYTE (MEM)	- calculate available program space
SAVE	- cassette programme save
LOAD	- cassette programme load
DATA]	- load and read data string
READ	
RESTORE	- resets data pointer
TOG (F/I MODE)	- sets math function
RND	- random number generator
TAB	- cursor control
math operators	- + - / *
math comparitors	- > < =

SIN, COS, TAN, SQUARE, absolute value, LOG, INTERGER

Unique Netronics Commands

DEVICE 0-3	- selects I/O device
SET K, P, Q	- select serial baud rate
EXAM	- list all variables in use
EXCL	- delete variable from program
INPORT	- input hex data via ports 1-7
OUTPORT	- output hex data via ports 1-7
STOP	- programable break point
CTL	- control-cursor command
HELP	- list command set

Unique Quest Commands

RUN (expr)	- run from selected line #
DIM (var)	- ARRAY
DSAVE	- cassette save data only
DLOAD	- cassette load data only
DFUS	- define user space start location
EOP	- prints address of end of program
EOD	- prints address of end of data
WAIT	- programable delay
CHRS	- print ASCII character
ASC (string)	- returns value of ASCII characeters in string
LEN "	- returns # of ASCII characters in string.

...Continued on Page: 52

Bill Eckel
7711 So 73 Avenue
Omaha, Nebraska 68128

ROLL OVER HEX WHILE LOADING

I am very limited in my programing experience. Never-the-less I am slowly learning. Your request for members to share their programs, no matter how small they are, because others will learn by them and eventually be able to submit their own or even improve upon them, is very true as seen by modifications to members programs already submitted by others. This is the case with me. In Ispo Facto #9 on page 9, Tom Jones submitted a little program on entering data with a rollover effect on the hex displays before pushing the input key. Although I have the Elf II with monitor system, I have found this program very useful. It will tell you if you depressed your keys completely or if the wrong information got entered before you load it into memory. I modified this program so that is may be entered on page one and load on any page in memory.

Address	Code	Mneumonics
00	F8__	LDI (set high address of page to loade (should be 00 if 1st page)
02	BF	PHI F
03	F8__	LDI (set low starting address of area to load. If on page 1, must be 12)
05	AF	PLO F
06	EF	SEX F
07	6C	INP 4
08	64	OOT 4
09	2F	DEC X
0A	3F 07	BN4
0C	6C	IMP 4
0D	64	OUT 4
0E	37 0E	B4
10	30 06	BR
12	STK if set for page 1	

Address 01 will be the high address of the page to load. Example: 00 for page 1, 01 for page 2, etc. Address 04 will be 00 if loading on any page but 1. If loading on page 1 (first 256 bytes), address 04 must be 12. After loading display and appear to shift to left. Pushing input will load data into memory starting at the address sent into the program. If program is set to load on page one (address 01 is 00 and address 04 is 12), after loading, reset computer and load the first 12 bytes as normal.

Perhaps some one can take this program and incorporate it with some existing monitor systems.

SOFTWARE FOR THE ETI 2708 EPROM PROGRAMMER

B. Murphy
102 McCraney St.
Oakville, Ont.

Some of you may have considered building the EPROM programmer that was written up in the December 1978 issue of ETI but hesitated due to the lack of software.

This article solves the 1802 software problem for you. The program to run the EPROM programmer is shown in listing 1--all 60 hex bytes of it.

HOW IT WORKS

Since the ETI programmer was designed to be a "universal" 2708 programmer, it uses a 300 BPS serial interface. The majority of the program involves serializing the data via the Q line.

HOW TO RUN THE PROGRAM

Set up the EPROM programmer UART for 8 data bits,no parity and 1 stop bit. Having connected the programmer via serial 20 ma. loop interface, serial RS232 interface, or just a 4050 buffer, key in the program on any memory page of the 1802. Next, load the 1K of data you want to program in any 4 consecutive memory pages. Call the program using R3 as the program counter. If you want, you can use R0 as the program counter by changing location 1 to a 90 and location 3e to a D0. Once running,the program expects you to enter the page number where the data to be programmed resides. Key in the page number,hold down the reset button down until the EPROM programmer is powered up. Next, hit the "I" key and go for supper...it takes about 50 minutes to program the 2708 at 300 bps!

POSSIBLE ENHANCEMENTS

You could speed things up by increasing the interface speed to 1200 or 2400 BPS. This could be done by re-adjusting the 555 timer to run at 1200x16 Hz or 2400X16 Hz respectively. The speed constant at location 52 would also have to be changed.

If you are only programming 256 bytes of information you need not program all 1024 locations. To do this, simply change location 21 to a 01.

Well, thats the software story--the ETI article is very complete on the hardware side.

Remember that if you decide to build the programmer do not skimp on the capacitors. You should only use high quality mylar capacitors in the timing circuits. Good luck!

References:

1. EPROM Programmer, Electronics Today, December 1978, p. 29
2. An RS232 Interface, IPSO FACTO, issue 5, p. 4

```
                     1 ;.....................................................
                     2 ;;
                     3 ;;      THIS PROGRAM IS DESIGNED TO PROGRAM A EPROM
                     4 ;;      VIA THE  Q LINE.
                     5 ;;
                     6 ;;        REGISTER USAGE:      R0-R1=FREE
                     7 ;;                             R2=STACK
                     8 ;;                             R3=MAINLINE P.C.
                     9 ;;                             R7=TX SUBR P.C.
                    10 ;;                             R8.0=# OF TIMES TO PGM
                    11 ;;                             R8.1=WORK
                    12 ;;                             R9=ADDR OF DATA
                    13 ;;                             R10=FREE
                    14 ;;                             R11.0=WORK
                    15 ;;                             R12=DELAY WORK
                    16 ;;                             R13=TX SUBR WORK
                    17 ;;                             R14=BIT COUNTER
                    18 ;;                             R15=DELAY P.C.
                    19 ;;
                    20 ;.....................................................
0000      =00FF     21 STACK     EQU   #00FF          ;;STACK ADDR
0000 E2             22           SEX   R2             ;;STACK IS R2
0001 93             23           GHI   R3             ;;GET CURRENT PAGE
0002 B7             24           PHI   R7             ;;SET UP R7
0003 BD             25           PHI   R13            ;;SET UP R13
0004 BF             26           PHI   R15            ;;SET UP R15
0005 F8FF           27           LDI   A.0(STACK)     ;;SET
0007 A2             28           PLO   R2             ;; UP
0008 F800           29           LDI   A.1(STACK)     ;;    THE
000A B2             30           PHI   R2             ;;      STACK
000B F83F           31           LDI   A.0(TXENT)     ;;SET UP TX ROUTINE
000D A7             32           PLO   R7
000E F857           33           LDI   A.0(DELAY)     ;;SET UP DELAY
0010 AF             34           PLO   R15
0011 7B             35           SEQ                  ;;Q TO MARK MODE
0012 3F12           36           BN4   *              ;;SET UP
0014 3714           37           B4    *              ;;AND READ
0016 6C             38           INP4                 ;;PAGE
0017 64             39           OUT4                 ;;WHERE DATA IS.
0018 B8             40           PHI   R8             ;;SAVE IT
0019 F87D           41           LDI   125            ;;# OF TIMES TO PGM 2708
001B A8             42           PLO   R8             ;;SAVE AWAY
001C F800           43 NEWCYC:   LDI   0              ;;CLEAR OUT
001E A2             44           PLO   R2             ;;STACK AND
001F A9             45           PLO   R9             ;;SIZE.
0020 F804           46           LDI   4              ;;SIZE OF PGM AREA
0022 B9             47           PHI   R9             ;;IS 1K BYTES.
0023 98             48           GHI   R8             ;;ADDR OF RAM AREA.
0024 B2             49           PHI   R2             ;;SET UP IN R2.
0025 F0             50 NXBYTE:   LDX                  ;;GET DATA BYTE.
0026 64             51           OUT4                 ;;DISPLAY & INC R2.
0027 D7             52           SEP   R7             ;;CALL OUTPUT ROUT.
0028 29             53           DEC   R9             ;;DEC BYTE COUNT.
0029 89             54           GLO   R9             ;;GET IT.
002A 3A25           55           BNZ   NXBYTE         ;;ARE WE DONE?
002C 99             56           GHI   R9             ;;NO CHECK HI BYTE
002D 3A25           57           BNZ   NXBYTE         ;;CHECK AGAIN
002F F8A0           58           LDI   #A0            ;;SET UP LOOP
```



```
0031 B8             59           PHI   R11            ;;TO RUN 1 SEC...
0032 2B             60 TIMOUT:   DEC   R11            ;;
0033 8B             61           GLO   R11
0034 3A32           62           BNZ   TIMOUT         ;;DONE?
0036 9B             63           GHI   R11
0037 3A32           64           BNZ   TIMOUT         ;;DONE ?
                    65 ;;
0039 28             66           DEC   R8             ;;OF TIMES TO PGM
003A 88             67           GLO   R8
003B 3A1C           68           BNZ   NEWCYC         ;;ARE WE DONE
003D 00             69           IDL                  ;;JUST SIT THERE
```

```
LOCN OBJ CODE  STMT    SOURCE STATEMENT      1802 VER 1.7A     PAGE    3

                 70 ..........................................................
                 71 ..         OUTPUT ROUTINE
                 72 ..
                 73 ..         REGISTER USAGE :      R15=PC FOR DELAY SUBR
                 74 ..                               R14=BIT COUNTER
                 75 ..                               R13=CHAR INPUT
                 76 ..                               R12=LOOP COUNTER
                 77 ..                               R7=PC FOR TXENT
                 78 ..
                 79 ..         INPUTS:               D=DATA TO BE SENT
                 80 ..
                 81 ..         OUTPUTS:              Q LINE...SERIAL OUT
                 82 ..
                 83 ..........................................................
003E D3          84 TXRTN:     SEP   R3              RETURN TO CALLER
003F AD          85 TXENT:     PLO   R13             SAVE CHAR IN D
0040 F808        86            LDI   8               8 BITS PER CHARACTER
0042 AE          87            PLO   R14             NUMBER OF BITS TO GO
0043 7A          88            REQ                   SET START BIT
0044 DF          89            SEP   R15             DELAY ONE BIT TIME
0045 8D          90 TXSHIFT    GLO   R13             GET BITS TILL NOW
0046 76          91            SHRC                  GET HIGH BIT INTO DF
0047 AD          92            PLO   R13             SAVE SHIFTED VALUE
0048 CF          93            LSDF                  TEST IF BIT WAS THERE
0049 7A          94            REQ                   NO...SO SET SPACE
004A 38          95            SKP                   SKIP THE SEQ
004B 7B          96            SEQ                   SET TO MARK
004C DF          97            SEP   R15             DELAY ONE BIT TIME
004D 2E          98            DEC   R14             NUMBER OF BITS TO GO
004E 8E          99            GLO   R14             INTO D
004F 3A45       100            BNZ   TXSHIFT         BRA IF NOT DONE YET
0051 7B         101            SEQ                   SET STOP BIT (MARK)
0052 DF         102            SEP   R15             DELAY FOR 1 STOP BIT
0053 DF         103            SEP   R15             ANOTHER ONE...
0054 303E       104            BR    TXRTN           RETURN TO CALLER
                105 ..........................................................
0056 D7         106 DLYRTN:    SEP   R7              BACK UP 1 LEVEL
0057 F83E       107 DELAY:     LDI   62              1 BIT TIME AT 1MGHZ
                108 ..NOTE:    ABOVE VALUE WORKS ONLY FOR 1 MHZ CLOCK
                109 ..         USE   HEX 71 FOR A 1.795 MHZ CLOCK
0059 AC         110            PLO   R12             SAVE AWAY
005A 2C         111 DECR:      DEC   R12             COUNT DOWN
005B 8C         112            GLO   R12             RESULT INTO D
005C 3A5A       113            BNZ   DECR            DONE ?
005E 3056       114            BR    DLYRTN          RETURN TO CALLER
                115 ..
0060            116            END

   0 DIAGNOSTICS GENERATED
  32 SYMBOLS

SYMBOL TABLE:
STACK    00FF    NEWCYC   001C    NXBYTE   0025    TIMOUT   0032
TXRTN    003E    TXENT    003F    TXSHIFT  0045    DLYRTN   0056
DELAY    0057    DECR     005A
```

Dear Bernie:

E.N. Shepard,
903 12th St.,
Huntsville, Tx.,
USA 77340

I surely do enjoy IPSO FACTO. I have an ELF II. Is anybody else having trouble getting Netronics disassembler to load? Mine won't load and Netronics refuses to answer my requests to re-record. Looks like I have been had!

Keep up the good work.

(Eugene: Hope someone out there can help. Have you checked the tape in case there is a second file/copy? -Editor)

Some Thoughts on Device-independent I/O

Volker Raab
Ramtenvy 30
DK8581 Nimtofte
Denmark

In issue 12, Wayne Bowdish presented a fine article about deviceindependent i/o with just one mistake. It's the same mistake that most of us software people, including myself, make most of the time -- he is aiming far too high.

Frankly, I must admit that I rather like his ideas, but we must not forget that our aim should be a powerful, but TINY i/o system. What Wayne does suggest is an essential part of an operating system. Of course, no one can suggest any meaningful future software systems without an operating system in mind. I feel that we should not invent an operating system in bits and pieces, and am not sure that the time has come to decide on THE 1802 operating system. On the other hand, Jim Howell has convinced me that an 1802 operating system should be a TINY UNIX system. I would like Wayne to discuss his ideas with Jim before he gets too far. There are, however, some principles I want to add to the discussion:

1. About operating systems.

Any system should be a tiny system, meaning that it should be based on a suitable collection of simple (but powerful) ideas.

2. Files and Directories.

Files can be placed anywhere, in RAM or PROM (must be a read-only file - obviously) on cassette tape or floppy disk. They may be entered subsequently from a keyboard or written to a printer or display (TV screen). A directory is just another file. A system directory contains all directly accessible files, i.e. the physical devices, or rather their descriptions. Another directory contains the working files, or rather their descriptions, including a reference to the physical device in case of multi-file devices. Other directories (on cassette or floppy) are updated to and from this working directory. This is one of the ideas of the UNIX system. One of its benefits is the fact that i/o handling is exactly the same thing as file handling as seen from a progam.

3. Programs.

Internally, programs use references (numbers) for the files which they are using. It is the task of the operating system (or operator) to connect a logical file of a program to a meaningful file of the current directory and thereby to a meaningful physical device, before invoking the program.

4. I/O as Seen from the Program.

A file is a sequence of bytes, and is identified by a unique identification in the program. A file may be divided physically into blocks or lines, and may be addressable (byte addressable for RAM or ROM file, block addressable for a floppy disk file).

What can a program do to a file?

a) read the next byte.
b) write the next byte.
c) increase, decrease, or set the byte counter, if the file is byte addressable.
d) increase, decrease, or set the block counter, if the file is block addressable.
e) read a block. -¦- i.e. initialize a block transfer, if
f) write a block. -¦ the file is divided into blocks.
g) open a file.
h) close a file.

And that's most of it. Whether a file is buffered, queued, or nothing at all, is of no concern to the program. Example: if the program issues a read-byte, the program may immediately receive a byte from a buffer, and continue, or it will have to wait until someone types a character on the keyboard. Obviously, a program should be able to ask any file about its status, and the actual values of its byte or block counters in order to be able to manipulate them.

Error recovery may require in some cases that the program provides a procedure that handles errors, but the normal situations should be:

```
  Input: 1. Open the file (for input?) (read first block?)
         2. Read all bytes one at a time.
         3. Close the file.
 Output: 1. Open the file (for output?)
         2. Write all bytes, one at a time.
         3. Close the file (write last block?)
 Update: 1. Open the file.
         2. Set the byte-counter.
         3. Read a byte.
         4. Decrease the byte-counter.
         5. Write a file
         6. Close the file.
```

All the work should be done by a file-handler or i/o manager, i.e. blocking, waiting for transfers, and so forth.
>

THE 1/Nth DEGREE

Bernie Murphy,
102 McCraney
Oakville, Ont., Canada,
L6H 3H6

A question by one of our readers prompted me to write this short article. His question was how to easily compute the cube root of a number using integer arithmetic a la TINY BASIC.

Going back in the memory banks, I recalled that it can be shown that the Nth root of a number A can be computed by the iterative formula:

$$X[n+1] = 1/N\ ((n-1)Xn + A/Xn^{**}(n-1))$$

for a square root, we have the familiar:

$$X[n+1] = 1/2\ (Xn + A/Xn)$$

The derivation of the above formula can be found in any good College Numerical Analysis text.

Taking Tom Crawford's article on A Tiny Basic Square Root Routine, IPSO FACTO #7, page 44 as a basis, I came up with the general solution as shown on the following page. Lines 1070 to 1075 evaluate Xn**(N-1) and line 1076 is the iterative formula. Some sample data values and the corresponding results as found following the program. High math strikes again!!!

Sample Program and Output:

```
50 PR'ENTER NUMBER AND ROOT'
100 INPUT A,N
200 GOSUB 1000
300 PR'THE ';N;'TH ROOT OF ';A;' IS ';B
400 GOTO 50
1000 REM NTH ROOT ROUTINE RETURNS NTH ROOT OF A
1010 REM IN B, WORKSPACE IS C,D,T,Q
1020 IF A>1 GOTO 1050
1030 B=1
1040 RETURN
1050 B=A/182+2
1060 D=0
1070 REM COMPUTE B**N-1
1071 T=N
1072 Q=1
1073 Q=Q*B
1074 T=T-1
1075 IF T <> 1 GOTO 1073
1076 C=((N-1)*B+A/Q)/N
1077 IF (A-A/(2*B)*B*2) > B C=C+1
```

```
1078 REM ABOVE TO SEE IF C SHOULD BE ROUNDED UP
1080 IF B=C RETURN
1085 REM CHECK FOR OSCILLATION OF RESULT
1090 IF (B-C) =D GOTO 1130
1100 D=C-B
1110 B=C
1120 GOTO 1070
1130 REM PICK LARGEST OF B OR C
1140 IF B>C RETURN
1150 B=C
1160 RETURN

:RUN
ENTER NUMBER AND ROOT? 49,2
THE 2TH ROOT OF 49 IS 7
ENTER NUMBER AND ROOT? 64,3
THE 3TH ROOT OF 64 IS 4
ENTER NUMBER AND ROOT? 100,2
THE 2TH ROOT OF 100 IS 10
ENTER NUMBER AND ROOT? 1024,8
THE 8TH ROOT OF 1024 IS 2
ENTER NUMBER AND ROOT? 55,2
THE 2TH ROOT OF 55 IS 7
ENTER NUMBER AND ROOT? 32767,2
THE 2TH ROOT OF 32767 IS 181
ENTER NUMBER AND ROOT? 27,3
THE 3TH ROOT OF 27 IS 3
```

>

8K EPROM Memory Board

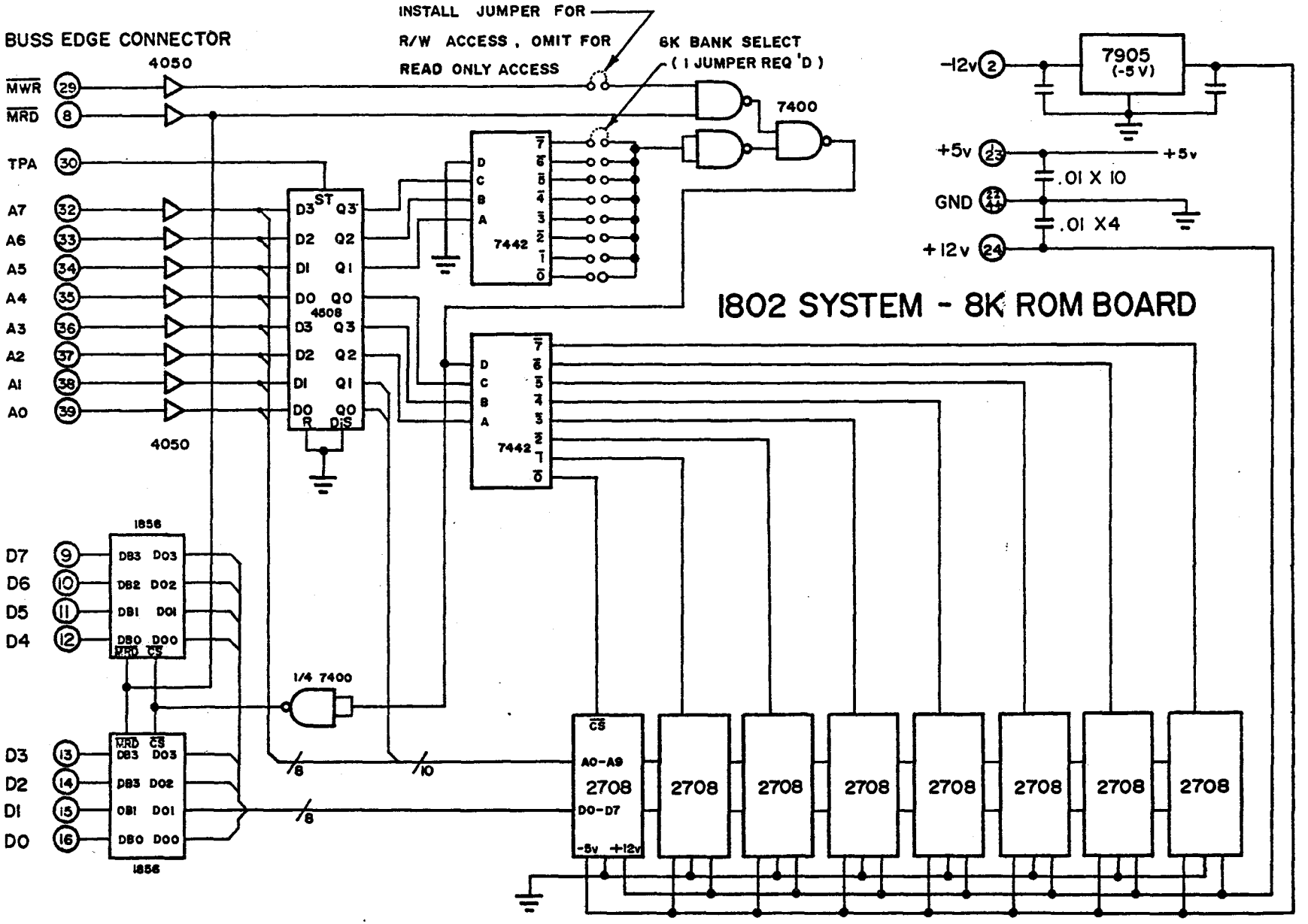
Tom Crawford
50 Brentwood Drive
Stoney Creek, Ont.
L8G 2W8

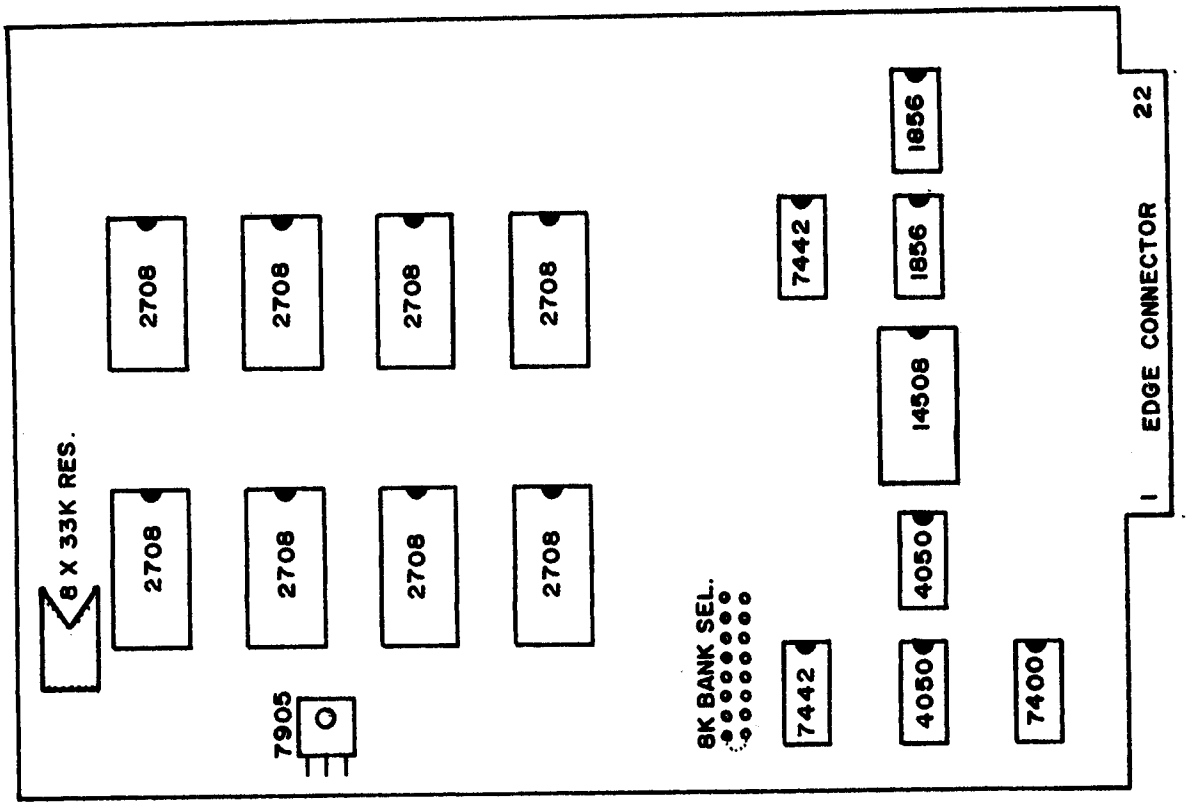
This printed circuit board was developed to meet the requirement of keeping system software (Monitor, Tiny BASIC, etc.) readily available in ROM. It is designed for the 44-pin bus used with the Tektron TEK-1802 micro-computer (See Figure 1), with several small modifications.

The board will hold up to 8 2708 EPROM chips. These chips occupy contiguous 1K areas in an 8K block of memory space. The 8K block can be selected as 1 of 8 blocks in the CPU°s 64K memory address space, using on-board DIP switches. The address and data lines are fully buffered (see the schematic, Figure 2).

Power requirements for the board are as follows:

Voltage		1K installed	8K installed
+5 volts	- typical	7 ma.	
	- maximum	10 ma.	
+12 volts	- typical	44 ma.	350 ma.
	- maximum	65 ma.	520 ma.
-12 volts	- typical	34 ma.	270 ma.
	- maximum	45 ma.	360 ma.


BUSS EDGE CONNECTOR
4050
INSTALL JUMPER FOR
R/W ACCESS , OMIT FOR
READ ONLY ACCESS
8K BANK SELECT
(1 JUMPER REQ 'D)
7400
MWR 29
MRD 8
TPA 30
A7 32
A6 33
A5 34
A4 35
A3 36
A2 37
A1 38
A0 39
4050
7442
4508
1856
D7 9
D6 10
D5 11
D4 12
D3 13
D2 14
D1 15
D0 16
1/4 7400
8
10
8
2708
AO-A9
DO-D7
-5v +12v
-12v 2
7905
(-5 V)
+5v 23
+5v
.01 X 10
GND 12 22
.01 X4
+12v 24
1802 SYSTEM - 8K ROM BOARD



COMPONENT LAYOUT FOR THE 8K ROM BOARD

Some Thoughts on the Future of the 1802

Bob Silcox
562 Forestwood Cres.
Burlington, Ont.
L7L 4K3

Various speculations on the future of the 1802 (and of A.C.E) have appeared in these pages, so this is not a new issue. However, it seems to me that it is a subject which could still stand some more thought. Below is a letter which was sent to our treasurer.

Ken Mantei
1254 E. 26th Street
San Bernardino, Calif.
92404

Dear Mike Franklin,

Thank you for explaining why I had not received IPSO FACTO since October. (Editor°s note - Mike explained to him that, among other reasons, the decreasing volume of good quality articles has caused delays in the appearance of some issues.) The problem of decreasing contributions by readers struck me, so I am returning a comment and suggestion.

The same decline has hit the Los Angeles Elf of the Valley club!! We have not published a newsletter since last summer. At our meetings it is clear that everyone is moving apart as each person grows and builds his own unique system. Many have purchased TRS-80°s too, or VIP°s, or modems to connect their terminals to subscription telephone timeshare systems.

Articles were getting so specialized that only one or two other members really got anything out of most of the construction articles. My interest moved to Assembly language and 2716 EPROM programming. I did a lot of work but didn°t feel it worthwhile to write up accounts of most of it. Oral presentations of my work at meetings elicited little response. I think this phenomenon may be widespread.

We watched IPSO FACTO and our Elf newsletter grow for two years. Then, when contributions fell off, we quit publishing and waited for enough material to accumulate to make a full-sized °respectable° issue. This may be a mistake.

I think the publication should continue, with a reduced number of pages if necessary. It°s important to keep the lag between contribution and publication small. When submitted articles don°t show up quickly, authors will stop writing. So I suggest IPSO FACTO stay on a bi-monthly schedule even if it is only 10 pages! It may experience a permanent size decrease. But if its appearance becomes irregular, I think it will die very quickly.

I acknowledge my share of the guilt. I have some ideas I°ve been meaning to write up--but just haven°t done it. Shame on me. I also want to thank you and the rest of the IPSO FACTO staff, past and present, for your efforts. I have really enjoyed IPSO FACTO. It is the main reason I went 1802 instead of buying a Z-80 starter kit.

Good luck,
Ken Mantei

In reviewing this letter, I must first acknowledge that the large gap between issues 14 and 15 was partly the fault of the editorial staff (it really is hard to get around to things during December, with Christmas shopping, parties, and so forth). The points he raises, however, are as relevant to A.C.E. as to the Elf of the Valley club.

Our articles have tended to become quite advanced and specialized both in the hardware and software areas, and while the response to the newsletter has been generally good, I can°t help wondering how much of most issues is really useful to most members.

A major root of these problems is the fact that a large number of our members have never expanded their systems beyond the 1K level, and so are unable to run the majority of the software which we have published. On the hardware side, people who do not have a technical background are often intimidated by the amount of knowledge necessary to get the hardware projects working properly.

We have not experienced as steep a fall-off in contributions as Mr. Mantei°s group has (knock on wood!!!), but the appearance of fairly advanced articles in IPSO FACTO seems to have had the effect of intimidating potential article writers whose projects are not on the same lofty level. This is really a pity, because it is precisely these more elementary-level submissions which are of use to the largest number, and which can help reduce instead of enlarging the gap (or should I say chasm) which seems to be appearing in the membership.

On the hardware side, club-sponsored projects such as the standard backplane, memory boards, and various other items, will, it is hoped, help a large number of our members to build their systems up to the point where they will have access to more of the material which we have published. This is a fairly necessary development, since I do not think that the club or newsletter can be viable for very long if we cannot present material requiring at least 8K memory without bypassing a majority of the membership. Applications for smaller systems have certainly not been exhausted (one area that comes to mind is use of micro°s to control household systems and appliances) but it would be very difficult to put together a whole newsletter, or even a significant portion of it dedicated solely to topics of use to systems in the 1K size area.

It hasn°t been my intention here to present a picture which is all gloom and doom. The possibilities for continuing success are there.

We must, however, address the problems raised here (and in Mr. Mantei°s letter) if we wish our long-term prospects to continue to be favourable.

Future Meetings

The March 11 meeting will be held at Sheridan College at 7:30 P.M. as previously announced.

However, the April 8 and May 13 meetings will be held back at our former location, the Stelco auditorium in Hamilton. These meetings will also begin at 7:30 P.M.
>

FOR SALE:

A.L. Van Caulart,
4102 Tree Top Crescent,
Mississauga, Ont.,
Canada. L5L 2L8
Tel: 416-828-7799 (Evenings)

1 ASR-33 TELETYPE with Tape winder
1 ASR-38 TELETYPE with Tape winder
1 KSR-38 TELETYPE

1 INTERDATA MOD 4 16 bit CPU with 32K bytes of memory together with NCR305 Cabinet, muffin fans and support rails. Architecture of this machine is near IBM/360 family.

1 IBM SYS/3 MOD 10 Computer System
1-5410 Processor 16K bytes
2-5440 Disk Drives (20 Meg/bytes, 10 fixed, 10 removeable)
1-5424 Multi-Function Card Unit (Capabilities of print/punch, interpret, merge sort, collate)
1-5413 200 LPM printer with 132 print positions
1-Set of IBM maintenance documentation with cart
1 Card Cabinet for 96 col cards
1 Wright Line Optiomedia Cabinet for disk packs
30 Disk packs for 5440 Disk Drives

----- Complete: Price only $20,000

Individual items can be negotiated.

Assembly Listing for COSMAC Graphics Program

In response to a number of requests, we are printing here the assembly listing to the graphics program for which object code was presented in the last issue.

It is hoped that this will make it easier to use, and will aid in overcoming any problems which might be encountered.

...Continued next page

```
                                THE START OF THE PROGRAM SECTION. **
                    ********************************************************************************
                    *
                    *       SUBROUTINE INIT
                    *
                    *
0400                        ORG     0400H
0400 FF             STACK   EQU     0FFH            ADDRESS WHERE STACK STARTS
0400 09             CMND    EQU     0700H           PAGE WHERE COMMANDS START TO BE READ OR WRITTEN
0400 09             ENDCOM  EQU     09F           AT THIS HIGH ORDER ADDRESS COMMANDS RECYCLE TO CMN
0400 C0 050D        SETUP   LBR     SETUP           LONG BRANCH USED TO AVOID OFFPAGE SHORT BRANCH
0403 F8 05          INIT    LDI#    INTRPT           HIGH ORDER INTERRUPT ADDRESS
0405 B1                     PHI     1
0406 F8 4D                  LDI#    INTRPT          LOW ORDER INTERRUPT ADDRESS
0408 A1                     PLO     1
0409 F8 09                  LDI     09H             CLEAR 256 BYTE PAGE (090C - 09FF)
040B BA                     PHI     0AH
040C F8 FF                  LDI     0FFH
040E AA                     PLO     0AH
040F EA                     SEX     0AH
0410 F8 00          CLEAR   LDI     00H
0412 73                     STXD
0413 8A                     GLO     0AH
0414 FE FF                  XRI     0FFH
0416 3A 10                  BNZ     CLEAR
0418 F8 05                  LDI#    DUMMY           HIGH ORDER ADDRESS OF DUMMY LOCATION
041A B7                     PHI     7                 IS PUT IN REGISTER 7
041C F8 CB                  LDI#    DUMMY           LOW ORDER ADDRESS OF DUMMY LOCATION
041E A7                     PLO     7                 IS PUT IN REGISTER 7
041F E7                     SEX     7
0420 69                     INP     1               TURNS ON 1861 T.V. CHIP
0421 F8 09                  LDI     09H             HIGH ORDER ADDRESS OF INITIAL CURSOR POSITION
0423 BE                     PHI     0EH             IS PUT IN REGISTER E
0424 F8 78                  LDI     078FH           LOW ORDER ADDRESS OF INITIAL CURSOR POSITION
0426 AE                     PLO     0EH             IS PUT IN REGISTER E
0427 F8 01                  LDI     01H             N=1
0429 AF                     PLO     0FH
042A 5E                     STR     0EH             FIRST BIT OF CURSOR BYTE =1
042B 7A                     REQ                     Q=0
042C F8 0C                  LDI     0CH
042E BF                     PHI     0FH             SET AUTO = OFF  (=0)
042F AC                     PLO     0CH              ERASE = OFF  (=0)
0431 BC                     PHI     0CH              JUMP = OFF  (=0)
0432 D5                     SEP     5               RETURN
                    *
                    ********************************************************************************
                    *       SUBROUTINE CURSOR
                    *
                    *
0433 E7             CURSOR  SEX     7
0434 F0                     LDX                     INPUT K GOES INTO D
0435 FD 0D                  SDI     0DH             K = D ?
0437 3A 3B                  BNZ     NOTD
0439 F8 01                  LDI     01H             SET AUTO ON (=1)
043B BF                     PHI     0FH
043C D5                     SEP     5               RETURN TO MAIN
043D F0             NOTD    LDX                     INPUT GOES INTO D
043E FD 0E                  SDI     0EH             K = E ?
0440 3A 46                  BNZ     NOTE
0442 F8 00                  LDI     00H             AUTO SET OFF (=0)
0444 BF                     PHI     0FH
0445 D5                     SEP     5               RETURN TO MAIN
0446 F0             NOTE    LDX
0447 FD 05                  SDI     05H             K = 5 ?
0449 3A 4F                  BNZ     NOT5
044B 7B                     SEQ                     Q = 1
044C F8 00                  LDI     00H             SET JUMP AND ERASE OFF (=0)
044E BC                     PHI     0CH
044F AC                     PLO     0CH
0450 D5                     SEP     5               RETURN TO MAIN
0451 F0             NOT5    LDX
0452 FD 00                  SDI     00H             K = 0 ?
0454 3A 5A                  BNZ     NOTO
0456 7A                     REQ
0457 F8 00                  LDI     00H             SET JUMP AND ERASE OFF (=0)
0459 BC                     PHI     0CH
045A AC                     PLO     0CH
045B D5                     SEP     5               RETURN TO MAIN
045C F0             NOTO    LDX
045D FD 0B                  SDI     0BH             K = B ?
045F 3A 67                  BNZ     NOTB
0461 7B                     SEQ
0462 F8 01                  LDI     01H             SET JUMP ON (=1)
0464 BC                     PHI     0CH
0465 F8 0C                  LDI     0CH             SET ERASE OFF (=0)
0467 AC                     PLO     0CH
0468 D5                     SEP     5               RETURN TO MAIN
0469 F0             NOTB    LDX
046A FD 0C                  SDI     0CH             K = C ?
046C 3A 7C                  BNZ     NOTC
```

```
                0460  F8 01       91          LDI     01H         ERASE SET ON (=1)
                0462  AD           92          PLC     0DH
                0463  D5           93          SEP     5           RETURN TO MAIN
                0464  0D           94  NOTC    GLC     0DH
                0465  3A 75        95          BNZ     ERASE       IF ERASE SET ON BRANCH TO ERASE
                0467  31 80        96          BQ      Q1          IF Q = 1 BRANCH TO Q1
                0469  D4           97  ERASE   SEP     4             CALL SUBROUTINE MASK
                046A  05 66        98          DW      MASK          C HAS MASK TO MAKE NTH BIT ZERO
                046C  FB FF        99          XRI     0FFH
                046E  EE           100         SEX     0EH
                046F  F2           101         AND
                0470  5E           102         STR     0EH
                0471  E7           103         SEX     7           RESET X TO 7
                0472  30 84        104         BR      JUMP
                0474  9D           105 Q1      GHI     0DH
                0475  3A 84        106         BNZ     JUMP          IF JUMP SET ON BRANCH TO JUMP
                0477  7A           107         REQ
                0478  F0           108 JUMP    LDX
                0479  FD 0A        109         SDI     0AH         K = A ?
                047B  3A 80        110         BNZ     NOTA
                047D  D4           111         SEP     4           CALL SUBROUTINE INIT
                047E  04 03        112         DW      INIT
                0480  D5           113         SEP     5           RETURN TO MAIN
                0481  F0           114 NOTA    LDX
                0482  FD 0F        115         SDI     0FH         K = F ?
                0484  3A 93        116         BNZ     NOTF
                0486  D5           117         SEP     5           RETURN TO MAIN
                0487  9F           118 NOTF    GHI     0FH         IF AUTO SET OFF BRANCH TO NAUTO2
                0488  32 9D        119         BZ      NAUTO2
                048A  F8 02        120         LDI     02H         TIME DELAY
                048C  BC           121         PHI     0CH
                048D  2C           122 DELAY   DEC     0CH
                048E  9C           123         GHI     0CH
                048F  3A 8D        124         BNZ     DELAY
                0491  F0           125 NAUTO2  LDX
                0492  FD 01        126         SDI     01H         K = 1 ?
                0494  3A A7        127         BNZ     NOT1
                0496  D4           128         SEP     4           CALL SUBROUTINE AREA1
                0497  05 3B        129         DW      AREA1
                0499  30 EE        130         BR      DOTV
                049B  F0           131 NOT1    LDX
                049C  FD 02        132         SDI     02H         K = 2 ?
                049E  3A B1        133         BNZ     NOT2
                04A0  D4           134         SEP     4           CALL SUBROUTINE AREA2
                04A1  05 4E        135         DW      AREA2
                04A3  30 EE        136         BR      DOTV
                04A5  F0           137 NOT2    LDX
                04A6  FD 03        138         SDI     03H         K= 3 ?
                04A8  3A BB        139         BNZ     NOT3
                04AA  D4           140         SEP     4           CALL SUBROUTINE AREA3
                04AB  05 53        141         DW      AREA3
                04AD  30 EE        142         BR      DOTV
                04AF  F0           143 NOT3    LDX
                04B0  FD 04        144         SDI     04H         K = 4 ?
                04B2  3A C5        145         BNZ     NOT4
                04B4  D4           146         SEP     4           CALL SUBROUTINE AREA4
                04B5  05 1D        147         DW      AREA4
                04B7  30 EE        148         BR      DOTV
                04B9  F0           149 NOT4    LDX
                04BA  FD 06        150         SDI     06H         K = 6 ?
                04BC  3A CF        151         BNZ     NOT6
                04BE  D4           152         SEP     4           CALL SUBROUTINE AREA6
                04BF  05 2C        153         DW      AREA6
                04C1  30 EE        154         BR      DOTV
                04C3  F0           155 NOT6    LDX
                04C4  FD 07        156         SDI     07H         K = 7 ?
                04C6  3A D9        157         BNZ     NOT7
                04C8  D4           158         SEP     4           CALL SUBROUTINE AREA7
                04C9  05 F2        159         DW      AREA7
                04CB  30 EE        160         BR      DOTV
                04CD  F0           161 NOT7    LDX
                04CE  FD 08        162         SDI     08H         K = 8 ?
                04D0  3A E3        163         BNZ     NOT8
                04D2  D4           164         SEP     4           CALL SUBROUTINE AREA8
                04D3  05 05        165         DW      AREA8
                04D5  30 EE        166         BR      DOTV
                04D7  F0           167 NOT8    LDX
                04D8  FD 09        168         SDI     09H         K = 9 ?
                04DA  3A ED        169         BNZ     NOT9
                04DC  D4           170         SEP     4           CALL SUBROUTINE AREA9
                04DD  05 0A        171         DW      AREA9
                04DF  30 EE        172         BR      DOTV
                04E1  D5           173 NOT9    SEP     5           RETURN TO MAIN
                04E2  D4           174 DOTV    SEP     4         CALL SUBROUTINE DOT
                04E3  C5 76        175         DW      DOT
                04E5  D5           176         SEP     5           RETURN TO MAIN
                                   177 *
                                   178 ********************************************************************************
                                   179 *       SUBROUTINE AREA7
                                   180 *
```

```
                                                *
04F2  3F          AREA7   GLC     0FH           D = N
04F3  FD 08               SDI     08H           N = 8 ?
04F5  3A FF               BNZ     NOT8V1
04F7  8E                  GLC     0EH
04F8  FF C9               SMI     09H
04FA  AE                  PLC     0EH
04FB  F8 01               LDI     01H
04FD  AF                  PLC     0FH
04FE  D5                  SEP     5             RETURN TO CURSOR
04FF  8E          NOT8V1  GLC     0EH
0500  FF 08               SMI     08H
0502  AE                  PLC     0EH
0503  1F                  INC     0FH
0504  D5                  SEP     5             RETURN TO CURSOR
                                                *
                                                ********************************************************************************
                                                *       SUBROUTINE AREA8
                                                *
                                                *
0505  8E          AREA8   GLC     0EH
0506  FF 08               SMI     08H
0508  AE                  PLC     0EH
0509  D5                  SEP     5             RETURN TO CURSOR
                                                ********************************************************************************
                                                *       SUBROUTINE AREA9
                                                *
                                                *
050A  8F          AREA9   GLC     0FH
050B  FD 01               SDI     01H
050D  3A 17               BNZ     NOT1V1
050F  8E                  GLC     0EH
0510  FF 07               SMI     07H
0512  AE                  PLC     0EH
0513  F8 08               LDI     08H
0515  AF                  PLC     0FH
0516  D5                  SEP     5             RETURN TO CURSOR
0517  8E          NOT1V1  GLC     0EH
0518  FF 09               SMI     08H
051A  AE                  PLC     0EH
051B  2F                  DEC     0FH
051C  D5                  SEP     5             RETURN TO CURSOR
                                                *
                                                ********************************************************************************
                                                *       SUBROUTINE AREA4
                                                *
051D  8F          AREA4   GLC     0FH
051E  FD 08               SDI     08H
0520  3A 2A               BNZ     NOT8V2
0522  8E                  GLC     0EH
0523  FF 01               SMI     01H
0525  AE                  PLC     0EH
0526  F8 01               LDI     01H
0528  AF                  PLC     0FH
0529  D5                  SEP     5             RETURN TO CURSOR
052A  1F          NOT8V2  INC     0FH
052B  D5                  SEP     5             RETURN TO CURSOR
                                                *
                                                ********************************************************************************
                                                *       SUBROUTINE AREA6
                                                *
052C  8F          AREA6   GLC     0FH
052D  FD 01               SDI     01H
052F  3A 39               BNZ     NOT1V2
0531  8E                  GLC     0EH
0532  FC 01               ADI     01H
0534  AE                  PLC     0EH
0535  F8 08               LDI     08H
0537  AF                  PLC     0FH
0538  D5                  SEP     5             RETURN TO CURSOR
0539  2F          NOT1V2  DEC     0FH
053A  D5                  SEP     5             RETURN TO CURSOR
                                                *
                                                ********************************************************************************
                                                *       SUBROUTINE AREA1
                                                *
053B  8F          AREA1   GLC     0FH
053C  FD 08               SDI     08H
053E  3A 4E               BNZ     NOT8V3
0540  8E                  GLC     0EH
0541  FC 07               ADI     07H
0543  AE                  PLC     0EH
0544  F8 01               LDI     01H
0546  AF                  PLC     0FH
0547  D5                  SEP     5             RETURN TO CURSOR
0548  8E          NOT8V3  GLC     0EH
0549  FC 08               ADI     08H
054B  AE                  PLC     0EH
054C  1F                  INC     0FH
054D  D5                  SEP     5             RETURN TO CURSOR
```

```
                                    *
                                    ********************************************************************************
                                    *       SUBROUTINE AREA2
                                    *
054E  8E            AREA2   GLO     OEH
054F  FC 08                 ADI     08H
0551  AE                    PLO     OEH
0552  D5                    SEP     5             RETURN TO CURSOR
                                    *
                                    ********************************************************************************
                                    *       SUBROUTINE AREA3
                                    *
0553  8F            AREA3   GLO     OFH
0554  FD 01                 SDI     01H
0556  3A 60                 BNZ     NOTIV3
0558  8E                    GLO     OEH
0559  FC 09                 ADI     09H
055B  AE                    PLO     OEH
055C  F8 08                 LDI     08H
055E  AF                    PLO     OFH
055F  D5                    SEP     5             RETURN TO CURSOR
0560  8E            NOTIV3  GLO     OEH
0561  FC 08                 ADI     08H
0563  AE                    PLO     OEH
0564  2F                    DEC     OFH
0565  D5                    SEP     5             RETURN TO CURSOR
                                    *
                                    ********************************************************************************
                                    *       SUBROUTINE MASK
                                    *
                                    *
0566  BF            MASK    GLO     OFH           COPY REGISTER F INTO REGISTER C
0567  AC                    PLO     OCH
0568  F8 01                 LDI     01H           D = 1
056A  BC                    PHI     OCH            HIGH ORDER REGISTER C = D
056B  2C            MASK1   DEC     OCH
056C  8C                    GLO     OCH
056D  32 74                 BZ      DZERO         D = 0 ?
056F  9C                    GHI     OCH            D = HIGH ORDER REGISTER C
0570  FE                    SHL                   SHIFT C LEFT
0571  BC                    PHI     OCH            HIGH ORDER REGISTER C = D
0572  30 6B                 BR      MASK1
0574  9C            DZERO   GHI     OCH            D=REG. C.1  NTH BIT = 1, OTHERS = 0
0575  D5                    SEP     5             RETURN
                                    *
                                    ********************************************************************************
                                    *       SUBROUTINE DOT
                                    *
                                    *
0576  D4            DOT     SEP     4             CALL SUBROUTINE MASK
0577  05 66                 DW      MASK
0579  6E                    SEX     OEH
057A  F1                    OR
057B  AC                    PLO     OCH
057C  F5                    SD
057D  3A 80                 BNZ     NZERO
057F  7B                    SEQ
0580  8C            NZERO   GLO     OCH
0581  5E                    STR     OEH
0582  F8 02                 LDI     02H           TIME DELAY
0584  BC                    PHI     OCH
0585  2C            DELAY1  DEC     OCH
0586  9C                    GHI     OCH
0587  3A 85                 BNZ     DELAY1
0589  D5                    SEP     5             RETURN TO CURSOR
                                    *
                                    ********************************************************************************
                                    *       SUBROUTINE BLINK
                                    *
                                    *
058A  F8 10         BLINK   LDI     10H           TIME DELAY WITH CHECK FOR INPUT
058C  BC                    PHI     OCH
058D  2C            DELAY2  DEC     OCH
058E  37 A4                 B4      KEY2
0590  9C                    GHI     OCH
0591  3A 8D                 BNZ     DELAY2
0593  D4                    SEP     4             CALL SUBROUTINE MASK
0594  05 66                 DW      MASK
0596  FB FF                 XRI     0FFH          SET NTH BIT TO  0
0598  EE                    SEX     OEH
0599  F2                    AND
059A  5E                    STR     OEH
059B  F8 10                 LDI     10H           TIME DELAY WITH CHECK FOR INPUT
059D  BC                    PHI     OCH
059E  2C            DELAY3  DEC     OCH
059F  37 A4                 B4      KEY2
05A1  9C                    GHI     OCH
05A2  3A 9E                 BNZ     DELAY3
05A4  D4            KEY2    SEP     4             CALL SUBROUTINE MASK
05A5  05 66                 DW      MASK
```

```
                05A7  EE          361             SEX     0EF             SET NTH BIT TO  1
                05A8  F1          362             OR
                05A9  5E          363             STR     0EF
                05AA  D5          364             SEP     5               RETURN TO MAIN
                                  365     *
                                  366     ********************************************************************************
                                  367     *
                                  368     *
                                  369     *       INTERRUPT ROUTINE
                                  370     *
                05AB  72          371     INTRTN  LDXA                    RESTORE D REGISTER
                05AC  70          372             RET                     RETURN FROM INTERRUPT
                05AD  22          373     INTRPT  DEC     2               REGISTER 2 IS STACK POINTER
                05AE  78          374             SAV                     SAVE T REGISTER ON STACK
                05AF  22          375             DEC     2
                05B0  52          376             STR     2               SAVE D REGISTER ON STACK
                05B1  C4          377             NOP                     SYNC
                05B2  C4          378             NOP                     SYNC
                05B3  C4          379             NOP                     SYNC
                05B4  F8 09       380             LDI     09H             PUT ADDRESS OF PAGE 9 INTO REGISTER 0
                05B6  B0          381             PHI     0
                05B7  F8 00       382             LDI     00H
                05B9  A0          383             PLO     0
                05BA  80          384     REFRSH  GLO     0               GET LINE START ADDRESS
                05BB  E2          385             SEX     2
                05BC  E2          386             SEX     2               NOP
                05BD  20          387             DEC     0
                05BE  A0          388             PLO     0
                05BF  E2          389             SEX     2               NOP
                05C0  20          390             DEC     0
                05C1  A0          391             PLO     0
                05C2  E2          392             SEX     2               NOP
                05C3  20          393             DEC     0
                05C4  A0          394             PLO     0
                05C5  3C BA       395             BN1     REFRSH          LOOP TIL "DISPLAY STATUS" LOW
                05C7  30 AB       396             BR      INTRTN
                05C9  C4          397             NOP
                05CA  C4          398             NOP
                05CB  00          399     DUMMY   DB      0H              DUMMY LOCATION USED BY REGISTER 7
                                  400     *
                                  401     ********************************************************************************
                                  402     *
                                  403     *
                                  404     *       CALL AND RETURN ROUTINES
                                  405     *
                05CC  D5          406     EXITA   SEP     5
                05CD  B8          407     CALL    PHI     08H             SAVE D
                05CE  E2          408             SEX     02H
                05CF  96          409             GHI     06H
                05D0  73          410             STXD
                05D1  86          411             GLO     06H
                05D2  73          412             STXD
                05D3  93          413             GHI     03H
                05D4  B6          414             PHI     06H
                05D5  83          415             GLO     03H
                05D6  A6          416             PLO     06H
                05D7  46          417             LDA     06H
                05D8  B3          418             PHI     03H
                05D9  46          419             LDA     06H
                05DA  A3          420             PLO     03H
                05DB  F8 CC       421             LDI     CALL            RESTORE P
                05DD  30 CC       422             BR      EXITA
                                  423     *
                                  424     *
                05DE  D3          425     EXITR   SEP     3
                05DF  B8          426     RET     PHI     08H             SAVE D
                05E0  96          427             GHI     06H
                05E1  B3          428             PHI     03H
                05E2  86          429             GLO     06H
                05E3  A3          430             PLO     03H
                05E4  E2          431             SEX     02H
                05E5  12          432             INC     02H
                05E6  72          433             LDXA
                05E7  A6          434             PLO     06H
                05E8  F0          435             LDX
                05E9  B6          436             PHI     06H
                05EA  98          437             GHI     08H             RESTORE D
                05EB  30 DE       438             BR      EXITR
                                  439     *
                                  440     ********************************************************************************
                                  441     *
                                  442     *       SETUP
                                  443     *
                05EC  F8 D5       444     SETUP   LDI#    CALL            SET UP REGISTERS 4 AND 5 FOR
                05EF  B4          445             PHI     04H             CALL AND RETURN ROUTINES
                05F0  F8 CD       446             LDI%    CALL
                05F2  A4          447             PLO     04H
                05F3  F8 D5       448             LDI#    RET
                05F5  B5          449             PHI     05H
                05F6  F8 DF       450             LDI%    RET
```

```
                05F8  A5          451         PRC     05H
                05F9  F8 06       452         LDI#    STACK       SET UP REGISTER 2 FOR THE STACK
                05FB  B2          453         PHI     02H
                05FC  F8 FF       454         LDI#    STACK
                05FE  A2          455         PLO     02H
                05FF  C0 0602     456         LBR     MAIN        W/ LONG BRANCH MAIN CAN BE WRITTEN ANYWHERE
                                  457 *
                                  458 *
                                  459 ********************************************************************
                                  460 *       MAIN
                                  461 *
                0602  D4          462 MAIN    SEP     4           CALL SUBROUTINE INIT
                0603  04 03       463         DW      INIT
                0605  F8 07       464         LDI#    CMND        ADDRESS OF PAGE WHERE COPY
                0607  B9          465         PHI     9           OF COMMANDS OCCS
                0608  F8 00       466         LDI#    CMND        (VIA REG. 9)
                060A  A9          467         PLO     9
                060B  3F 12       468 NKEY    B4      KEY         BRANCH IF KEYPAD DEPRESSED
                060D  D4          469         SEP     4           CALL SUBROUTINE BLINK
                060E  05 8A       470         DW      BLINK
                0610  30 0B       471         BR      NKEY
                0612  9F          472 KEY     GHI     OFH
                0613  3A 17       473         BNZ     ALTG        IF AUTO ON BYPASS CHECK FOR DEPRESSED KEYPAD
   1            0615  37 15       474 NAUTO   B4      NAUTO
                0617  E7          475 ALTL    SEX     7
                0618  6C          476         INP     4           READ INPUT FROM KEYPAD
                0619  FA 0F       477         ANI     OFH
                061B  57          478         STR     7           INPUT GOES INTO M(R(7))
                061C  FE          479         SHL                 SHIFT D LEFT 4 TIMES
                061D  FE          480         SHL                 TO PACK INPUT
                061E  FE          481         SHL
                061F  FE          482         SHL
                0620  59          483         STR     9           SHIFTED INPUT PUT INTO M(R(9))
                0621  D4          484         SEP     4           CALL SUBROUTINE CURSOR
                0622  04 31       485         DW      CURSOR
                0624  07          486         LDN     7
                0625  FD 0F       487         SDI     OFH         INPUT=F?
                0627  32 48       488         BZ      BEGIN
                0629  3F 3C       489 NKEY1   B4      KEY1
                062B  D4          490         SEP     4           CALL SUBROUTINE BLINK
                062C  05 8A       491         DW      BLINK
                062E  30 29       492         BR      NKEY1
                0630  9F          493 KEY1    GHI     OFH
                0631  3A 35       494         BNZ     AUTO1
                0633  37 33       495 NAUTO1  B4      NAUTO1
                0635  E7          496 ALTU1   SEX     7
                0636  6C          497         INP     4
                0637  FA 0F       498         ANI     OFH
                0639  57          499         STR     7           NEXT INPUT STORED IN M(R(7))
                063A  E9          500         SEX     9
                063B  F4          501         ADD
                063C  59          502         STR     9           INPUT PACKED INTO M(R(9))
                063D  19          503         INC     9           INCREMENT ADDRESS IN REG. 9
                063E  D4          504         SEP     4           CALL SUBROUTINE CURSOR
                063F  04 31       505         DW      CURSOR
                0641  07          506         LDN     7
                0642  FD 0F       507         SDI     OFH         INPUT=F ?
                0644  32 48       508         BZ      BEGIN
                0646  30 0B       509         BR      NKEY
                0648  F8 07       510 BEGIN   LDI#    CMND        PUT ADDRESS OF PAGE WITH STORED COMMANDS IN REG.
                064A  B9          511         PHI     9
                064B  F8 00       512         LDI#    CMND
                064D  A9          513         PLO     9
                064E  09          514 NEXT    LDN     9
                064F  F6          515         SHR                 SHIFT RIGHT 4 TIMES TO UNPACK COMMANDS
                0650  F6          516         SHR
                0651  F6          517         SHR
                0652  F6          518         SHR
                0653  57          519         STR     7           COMMAND PUT INTO REG. 7
                0654  D4          520         SEP     4           CALL SUBROUTINE CURSOR
                0655  04 31       521         DW      CURSOR
                0657  07          522         LDN     7
                0658  FD 0F       523         SDI     OFH         COMMAND = F ?
                065A  32 48       524         BZ      BEGIN
                065C  09          525         LDN     9
                065D  FA 0F       526         ANI     OFH
                065F  57          527         STR     7           NEXT COMMAND PUT INTO REG. 7
                0660  D4          528         SEP     4           CALL SUBROUTINE CURSOR
                0661  04 31       529         DW      CURSOR
                0663  07          530         LDN     7
                0664  FD 0F       531         SDI     OFH         COMMAND = F ?
                0666  32 48       532         BZ      BEGIN
   1            0668  19          533         INC     9
                0669  99          534         GHI     9
                066A  FD 09       535         SDI     ENDCOM      AT END OF STORAGE SPACE GO TO BEGIN
                066C  32 48       536         BZ      BEGIN         END OF STORAGE SPACE IS CONSIDERED AS
                066E  30 4E       537         BR      NEXT          BEGINNING OF DISPLAY PAGE
                                  538 *
                                  539 *
                                  540 *
```

PROTO BOARD

Eugene E. Jackson
3637 Snell Ave #385
San Jose. Calif
95136
ph: 408-224-0837

Last year I published a small article in Questdata mag. in an effort to form a local 1802 club in this so-called silicon valley. I received only 3 responses from people too far away to make a meeting practical. Even though I was dissappointed with the first attempt I will be willing to have my name, address, & phone number published in IPSO FACTO with hope that there are now enough 1802 fans nearby to form a club.

I like the idea of the project boards using the 44 pin buss. Quest made a design error by not buffering data befor hitting their 44 pin but this can be remedied by immediately buffering data on the interface adaptor that I will have to use before rearranging the pin assignments to yours. If one wants to get fancy, bells and lights can be added at this point for debugging purposes. May I suggest someone draw a sketch of this proposal for the next issue.

I have taken considerable time to provide as much detail as possible on this project in hope that someone may wish to try it. The sound generator chip was used to test the feasibility of the control board. This interface board was really the main goal of the project! As an avid experimenter hobbiest this board provides a means of evaluating hardware and theories. The SN76477 is a bargain and a compromise and if one wants to produce more exact effects there are better more expensive ways to do it.

A large gap seems to exist between the versitile micro and the real world due to lack of standards. Until the manufacturers can get their acts together the home micro will remain just an expensive toy.

There is one thing to consider as we wire our circuits, play the games and program our micros. We are evolving a new form of life. I hope some improvements can be made.

While I think of it, have any members had experience relating micros with handicapped or bedridden people. I ask because this is part of the reason I got involved in electronics as a child.

There is a vast resource of talent in such people if a few were willing to take the time to teach programming and theory.

I'll get back on the project now; sorry; sometimes I tend to wander and wonder.

As to the Proto Board questions, yes, several other chips and designs could accomplish the same thing but I designed this for maximum flexability with the least hardware. I think you will find

the extra work well worth the effort!

I have made certain important changes in the program & design that make the board easier to live with. Namely the 63 instruction (NØ + N1) is now used to strobe data to the board. This prevents the usual 64 output instruction from falsly strobing a D & A. I am including specific details about the board in the event you may publish or modify the design.

The 4515 decoder was chosen so that the extra 7 output lines could be used to select notes if a 50240 chromatic tone generator is added. I have an idea how to use the Hexkeypad for 8 octave note selection by direct entry, no conversion charts necessary, but more about that some other time.

The 4042 Q outputs were selected so that during a program loop that decrements a register the successive data will result in an increasing frequency. Note that 50 toggles faster than 5F. This can be inverted by replacing the 4049 buffer with a non-inverting 4050 type but be wure to switch to Q outputs on DD # 1 & 2 for active High logic.

The active High outputs of NØ, N1 & TPB are combined through a diode "and gate"; inverted and applied to the 4515 decoder to provide the data accept strobe pulse for the 4042 latches.

1 M ohm range trim pots may be added to the common leads of the D & A outputs which also presents a minimum value when all outputs are active High.

Two D & A's may be combined to form a single 256 step unit if the resistor matrix is changed but each D & A must be addressed by separate program bytes (5F + 67 = step 247 or 15 X 16 + ') 7).

Once the logic & parameters have been set for the SN76477 the only thing needed is the q output trigger pulse to activate the sound. This facilitates a wide variety of user programs since once the parameters have been latched all that is needed to produce sound is the Q trigger. As you see, one doesn't really need the 1802 at all; it mearly adds speed and variety to the sounds one wishes to produce. A few switches and a push button could do as well but much slower.

The little gadget wired to pin 11 of the SN76477 is a homemade optoisolator. Just take an LED & a photoresistor, put them together using a bit of heat shrink tubing to exclude outside lignt and that makes a voltage dependent resistor. I use it to modulate the output level by varing Q duty cycle but it could also be used to alter some other parameter as well. Fairly fast attack/decay periods can be implememted using the optoisolator under program control.

Radio Shack supplies an excellent data sheet on the SN76477 and I have tried to design around their parameters.

The program I have included is a modified version of "Four Byte Delight" with an 8 Byte D & A setup sequence tacked to the beginning. It references to page ØF for the sequence tables and includes the 63 output strobe feature. It you want to reference some other page change the byte at ØØ17 to the page desired. I use page ØF because of chip 8.

The short program at ØØ9Ø is independent of the first program and can be used by itself to check out the different byte effects.

Run this test program, then enter the sequence: 12,28,3C 4E,5A,6Ø,7Ø,8E via hexheypad using "I" key entry between bytes of course. Presto! Elf becomes R2D2. Now every "I" operation produces a new sound. To checkout a new value simply enter the new byte and operate "I" again. Example: to lengthen the decay period "8E", enter "8F" and operate "I". Be sure to write down what you have entered for parameters because this simple program only shows the new entry and the resulting sound depends upon all the previous entries as well.

Now try a test of the main program. Enter 3Ø1Ø and load the following sequences on page ØF starting at ØF ØØ:

R2D2 Versus The Telephone

ØF ØØ	12	28	3C	4E	5A	6Ø	7Ø	8E
	8Ø	Ø4	11	11	4F	Ø1	FØ	Ø8
ØF1Ø	5Ø	Ø4	ØØ	Ø1	8F	ØØ	ØØ	ØØ
	5F	ØØ	ØØ	ØØ	4Ø	ØØ	ØØ	ØF
ØF2Ø	2D	Ø3	11	11	5C	Ø3	FØ	ØF
	2Ø	Ø1	ØØ	Ø4	5C	Ø3	FØ	ØF
ØF3Ø	2A	Ø1	ØØ	Ø4	2Ø	ØØ	ØØ	ØØ
	28	Ø2	11	11	4E	ØØ	ØØ	ØF
ØF4Ø	5Ø	Ø4	11	11	2D	Ø6	11	11

The sequence tells a little story in sound. If you have used the same component values and wired your board like mine the message will be quite plain and is good for a laugh.

It's hard to believe that such unusually complex sounds can be created using just one chip.

If no sound results from a particular sequence suspect either a "2F" inhibit has been latched or the one shot parameter is too fast for the attack period.

A chart of the parameters is included to help in custom designing a particular sound.

I think I will wire this circuit as a permanent part of my Elf to replace the click/buzz output of the speaker. Right now a small transistor radio with a phono jack wired to the volume pot provides my audio amp.

>

BYTE CONTROL #84

Eugene Jackson
3637 Snell Ave., #385
San Jose, Calif., 95136

This is a control program suitable for use with the sound effects chip, the SN76477. Included in this article is the circuit and program for my 44 pin experimentor's board. The board incorporates a small Super Strip to accomodate any configuration of IC or parts. All inputs and outputs are tied to Molex type pins to facilitate patching via small jumpers to this strip. This way, the user can assign functions of hardware. The board could include 8 more D/A converters but space didn't permit.

Rather than flowcharts, the listings have been presented with a number of symbols.

Underlined bytes are relative jumps with an arrow indicating direction. The numbers indicate the LSD of the jump address. A bracket under a byte indicates a byte altered within the program

0072 85 B3 Save LR5 (pause) in HR3 (slow p.)

that can be modified to produce other effects.

I like to start programs at 0010 to leave room for expansion, so be sure to use '30 10' for access. An end-of-page halt at 0082 provides a gate to some other program.

Program Narrative:

The program starts at location 0010, occupies a half-page of memory, and takes data sequentially from tables on page 0F. These tables preset 8 D/A converters and very Q on and off duty cycles. The D & A and Q outputs set parameters on an experimental 44-pin expansion board.

Parameters can be defined as logic levels, D to R or D to C values on a Super Strip socket. Q functions as an overall output level control and trigger.

The table location on page 0F is selected via keypad entry and is displayed on the hex D output. Upon operation of the 'I' key, the program jumps to the selected location and displays the first 8 bytes from the table in sequence. This sets and latches each D/A converter on the '64' instruction. The program then looks at the following data in 4 byte sections. The first byte of this data is displayed on the hex display. If 00 is encountered here, the program returns for a new location to be specified. The second byte sets the number of repetitions, and the third sets Q->1 time and pause duration. The fourth sets Q->0 time and pause. If, for example, data continues for sixteen bytes, four separate sequences will occur with new data displayed with each four-byte set. Upon

completion of the sequences (00 detected), the program returns to display the last location selected. This enables the same sequence to be repeated by just operating the 'I' key. If a new table is desired, merely enter the location and push 'I'. Any number of separate functions or devices can be software controlled using this method.

Program Listing:

Address	Code	Comments
0010	90 BB AC BD	Set RB.1, RC.0, RD.1 to 00
14	F8 1B AD	Define insert point in RD.0
17	F8 0F BA	Define work page in RA.1
1A	F8 00 AA	Access to RA.0 for location insert
1D	ED	Set index pointing to data in RD
1E	6C 64 AA	Keypad entry (loc.), display, put RA.0
21	2D	Prevent advance of RD
22	3F 1D	Return for new entry if I = 0
24	37 24	Hold loop on I = 1
26	F8 08 AB	I = 0, now store seq# in RB.0
29	F8 40 BC	set display timer to 40 in RC.1
2C	EA	point X to data in RA
2D	64 2B	Display data, advance RA, decr. RB
2F	2C 9C 3A 2F	Decr. timer, return if not = 0
33	8B	time up, so check sequence count
34	3A 29	if not = zero, reset timer, get next byte

Sbr. Selects starting point on Pg 0F with keypad entry. Jumps to selected location and displays the first 8 bytes in succession for (40) duration.

* The sequence # may be any value and is always followed by the 4 Byte Sbr.

The timer SBR was added just to give a slow visual display of the 8 byte sequence.

0036	F0	Load current Data via RA to D
37	32 1D	If current D=00 return & display last location entry in RD, wait for new keypad entry.
39	64	If current Data .NOT.= 00 (1st byte) display and advance RA to (2nd byte)
3A	72	Load current Data via RA to D and advance RA to (3rd byte).
3B	A6	Copy current Data to LR6
3C	16	Increment R6 by 1 to assure 01 00 does not become FF on a Dec instruction
3D	72 A1 A5	(RA on 3rd byte) Save 3rd byte in LR and LR5
40	FA F0 A1	and imm. D with F0 to cancel the Least significant Digit and Str. in LR1
43	11	Inc R1 to assure 01

```
   44      85 FA 0F          Copy LR5 to D and AND IMM with 0F to
                             cancel the Most Sig. Digit
   47      FE FE FE FE       Shift D left 4x to recover as MSD
   4B      A5 15             Copy to LR5 and increment R5 to assure
                             01.
```

Sbr. Selects 4 Byte sequwnce if .NOT.=00 and disassembles the byte into nibbles. These are restored in the original registers after being processed.

```
  004D     90 A4             Set LR4 to 00 to initialize
  (4F)     81 A3             Copy LR1 to LR3
  (51)     23 83 3A 51       Dec. R3 and loop until = 00 (Q=0 times)
   55      39 5A             If Q=0 and R3=0 jump to (Q to 1)
   57      7A 30 5B          otherwise Q->0 goto R4 timer
  (5A)     7B                Q->1
  (5B)     14 84 A2          Inc R4 and  copy to LR2
  (5E)     22 82 3A 5E       Dec R2 and loop until=00 (Q=1 time)
   62      39 67             if Q=0 goto (Q to 1)
   64      7A 30 68          Q->0 goto R1 timer
  (67)     7B                Q->1
   68      21 81 3A 50       Dec R1 and loop until = 00
   6C      39 71             if Q=0 goto (Q to 1)
   6E      7A 30 72          Q->0 goto pause timer now
  (71)     7B                Q->1 start pause timer Sbr.
```

If Q enters this Sbr. as 0 it leaves as a 1 and v. versa.

Sbr. transfers Data through R1,2,3,4 making Q=1 or 0 decisions to achieve and incrementing Q->1 or decrementing Q->0 duty cycle. DATA ~08 gives short D. cycle. DATA ~F7 = long duty cycle.

```
  0072     85 B3 *           Save LR5 (pause) in HR3 (slow p.)
   74      23 93 * 3A 74     Dec R3 and loop until=00 (p. timer)
   78      31 3D             If Q=1 Q attack and pause is complete
                             (3rd byte) return to get Decay and pause
   7A      26 86 32 82       Dec R6 (Rep. value) if =00 goto Pg check
   7E      2A 2A 30 3D       (Rep .NOT.=00) then backspace RA 2 bytes
  (7D)                       and return to reinitiate Att. P. cycle.
   82      8A 3A 36          LRA .NOT.=0 nto end pf pg return to
                             start next 4 Byte sequence
   85      00 00 00          End of Page now Halt
```

Sbr. determines pause time, repeat sequence and end of page.

* sequences may be speeded up by shortening the pause timer and loading and checking LR3.

To use: Sequence all memory to 00 (90 AE BE EE 73 30 03)
------ Enter the program in memory as shown from 0010 to 0087

Enter an 8/4 byte sequence on Pg 0F as shown in the the example (location 0F20 to 0F2B)
Reset and enter a jump to the program using a 3010
Reset and run; enter 20 on the Hexpad and press "I" key. Program will jump to 0f20 and do the sequence. Operating "I" upon completion of the sequence will cause a repeat provided the same Location is selected.

```
Address  Code                                  Comments
0010     90 BB AC BD                --|
  14     F8 1B AD                     |  Insert Point on Keyboard entry
  17     F8 0F BA F8 00 AA            |  Work Pg and Loc insert Display
  1D     ED 6C 64 AA 2D 3F 1D       --|  location.
  24     37 24                      --|
  26     F8 08 AB                     |
  29     F8 40 BC                     |  Initial D/A Set (8 Seq)
  2C     EA 64 2B                     |  D/A Set timer.
  2F     2C 9C 3A 2F                  |  Display 1st 8 bytes.
  33     8B 3A 29                   --|
  36     F0 32 1D 64                --|
  3A     72 A6 16                     |  Byte disassembly and
  3D     72 A1 A5                     |  restore.
  40     FA F0 A1 11                  |
  44     85 FA 0F FE FE FE FE A5 15 --|
  4D     90 A4 81 A3                --|
  51     23 83 3A 51 39 5A            |
  57     7A 30 5B 7B                  |
  5B     14 84 A2                     |
  5E     22 82 3A 5E 39 67            |  Q->1  Q->0
  64     7A 30 68 7B                  |  Vary duty cycle.
  68     21 81 3A 50 39 71            |
  6E     7A 30 72 7B                --|
  72     85 B3 23 93 3A 74 31 3D    --|  Pause timer
  7A     26 86 32 82                  |
  7E     2A 2A 30 3D                  |  Backup 2 bytes
  82     8A 3A 36                     |
  85     00 00 00                   --|  end of page Halt.
```

Register assignments:

HRA	WORK REG
LRA	LOCATION INSERT POINT
LRB	D/A SETUP SEQUENCES
HRC	D/A HOLD AND DISPLAY TIME
LR1	INITIAL VALUE
LR2	SCRATCH FOR Q ON DURATION
LR3	SCRATCH FOR Q OFF DURATION
LR4	SCRATCH FOR A AND D VALUES
LR1	ATTACK OR DECAY VALUE SAVE
LR5	PAUSE VALUE SAVE
LR6	REPEAT VALUE

EXAMPLE OF A SEQUENCE: (Noise and one shot / 3 pauses)

```
0F20   12 22 38 40 50 61 7A 86     --> Initial display sequence
                                       (PRESETS 8 D/A'S)
0F28   AA 02 86 42      --> Q TRIGGER OUTPUT
       -- -- || ||
       |  |  || ||
       |  |  || |+---------Q=0 (pause on Q=0)
       |  |  || +----------Q->0 dur (decay speed)
       |  |  |+------------Q=1 dur (pause on Q=1)
       |  |  +-------------Q->1 dur (attack speed)
       |  +----------------2 repeats (3 cycles total)
       |                     Note: 00 here causes pgm to return
       +--------------------Index Byte (can be used to alter the effect)
```

(All FF on page 0F will take about 28.6 hours to complete
256-8 = 248 bytes divided by 4 = 62 seq * 256 rep *6.5 sec
=103,168 seconds!)

DD/1 ENVELOPE SELECT:

LOGIC INPUTS

(Byte #)	(4)	(2)		(1)
Data	Pin 28	Pin 1		Pin 3
10	0	0	VCO	External N input
12	0	1	One shot	
14	1	0	Mixer	
16	1	1	VCO/Act	

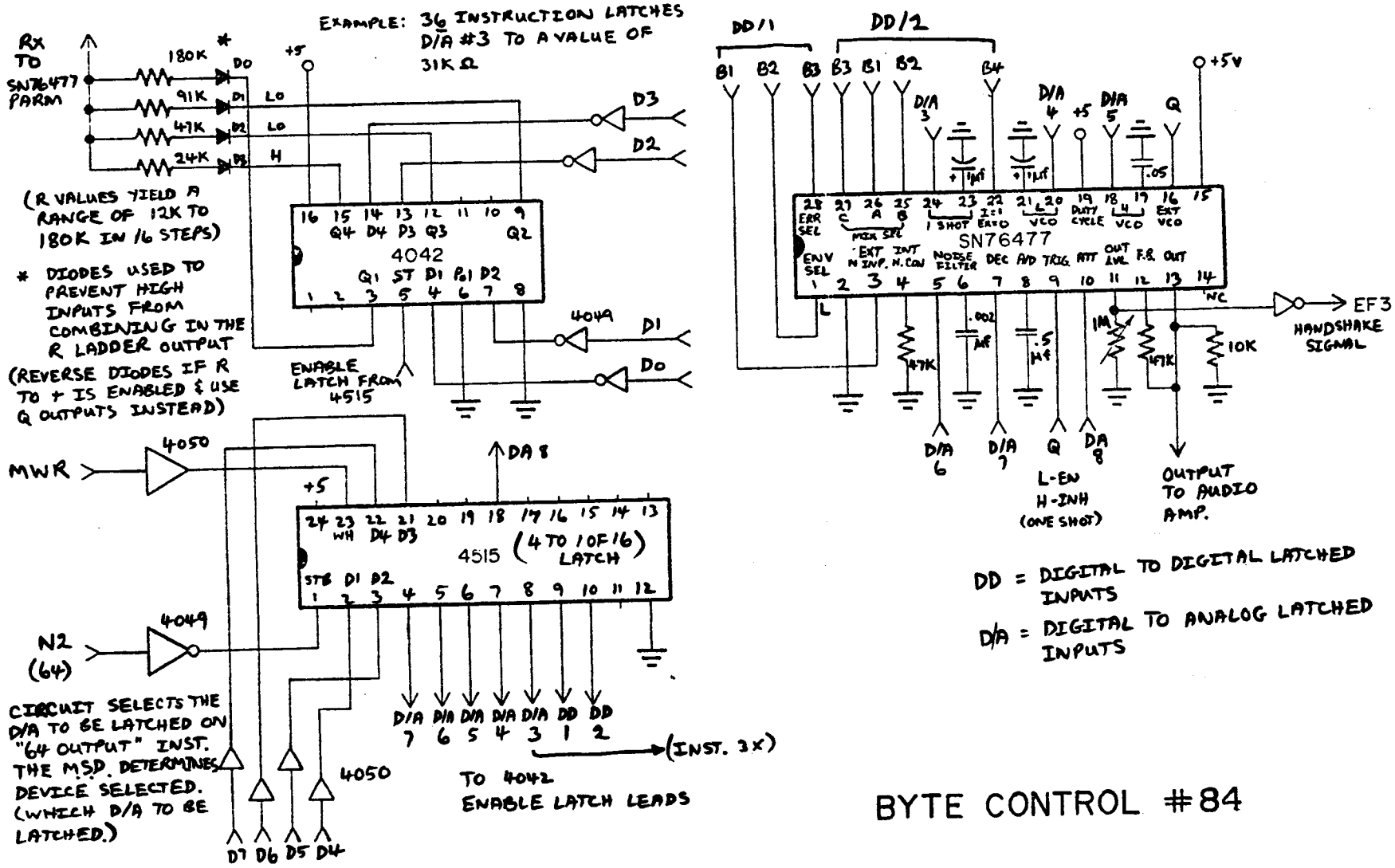
DD/2 MIXER SELECT:

(Byte #)	(4)	(2)	(1)		(8)
Data	Pin 27	Pin 25	PIN 26		Pin 22
20	0	0	0	VCO	1 = INT 0=EX
21	0	0	1	SLF	
22	0	1	0	N	
23	0	1	1	VCO/N	
24	1	0	0	SLF/N	
25	1	0	1	SLF/VCO/N	
26	1	1	0	SLF/VCO	
27	1	1	1	INHIBIT	

Parameter Test Program:

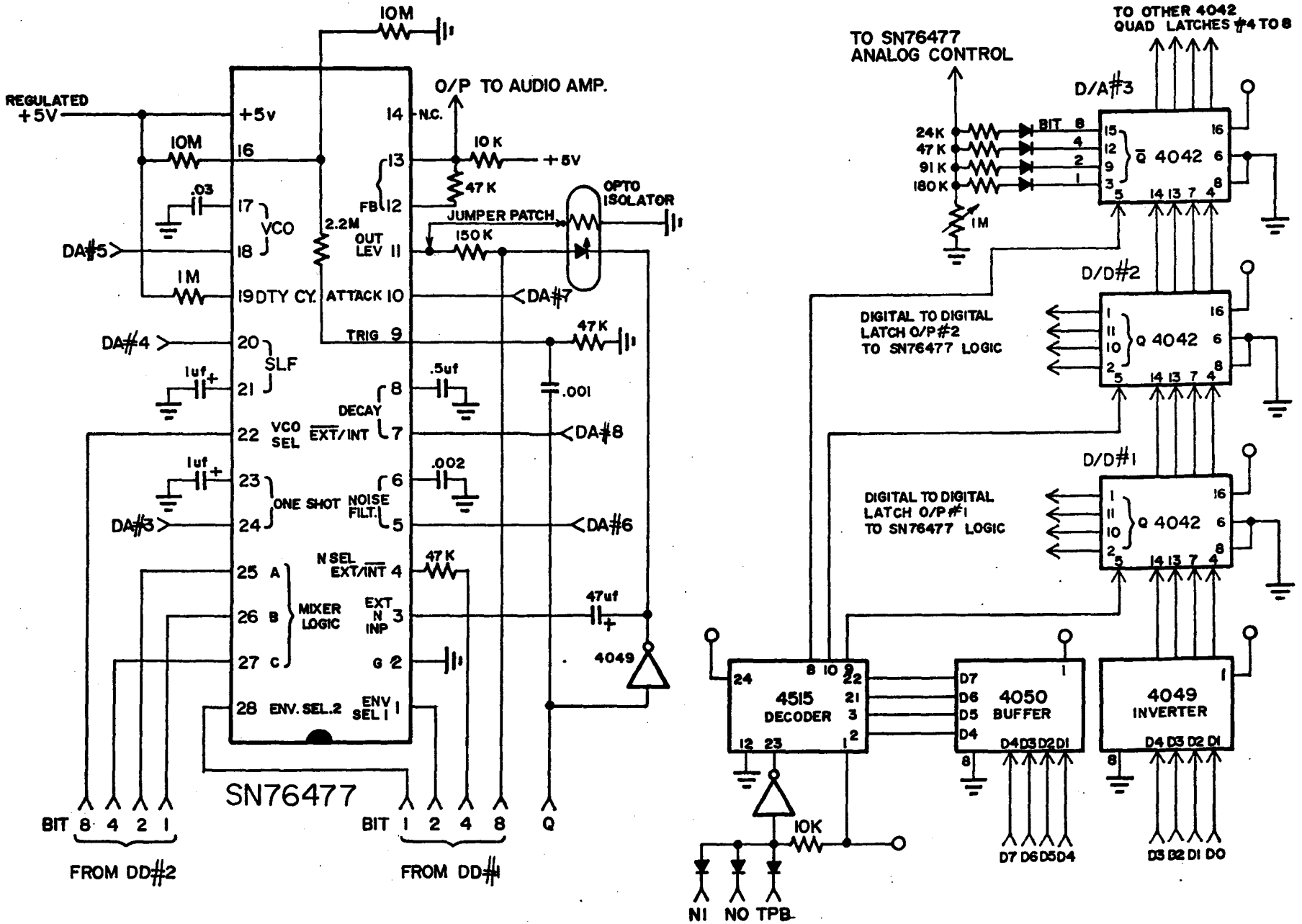
```
0000  7A E2 6C 64 22 3F 00          Display  keypad entry
  07  7B F8 FF B5 25 95 3A 0B      Q->1 Pause on I=1
  0F  37 08 30 00                  Return on I=0
```

Example: If on the keyboard a 36 is output, D/A #3 is selected and latched to level 6.



BYTE CONTROL #84

BYTE CONTROL #84



MINUTES OF MEETING - NOV. 13, 1979. **Mike Franklin**

Eugene and Anthony Tekatch and John Myszkowski prsented new products from Tektron for the 1802. Eugene presented a positive reportof the acceptance by industry of the 1802, and highlighted some of the remote sensing and multi-processor controlling applications currently under development. Anthony demonstrated the prototype of the NS57109 Number Oriented Microprocessor (NOM Dec,78, Jan 79 Radio Electronics) and John presented his MC6847 colour video board proto-type. Eugene also presented his company's versatile EPROM programmer.

The Executive were directed during the business portion of the meeting, to investigate methods or incorporation of the club, and to review alternatives, such as liability insurance, and to report the results at the next meeting. The club members in attendance endorsed the ACE hardware proposal and buss pin assignment, with further discussion to be held at the December meeting.

MINUTES OF MEETING - DEC. 11, 1979.

This meeting was devoted to club business, the subject of club incorporation and the hardware project. Mike Franklin presented a review of information supplied by Netronics on new products to be introduced soon. These products include a 10 slot 86 pin expansion board, an Epromer, Eprom board, A-D, D-A board and new software support.

With active interest evident from various manufacturers in the production of more 1802 products, the club discussed the merit of producing competing products, trying to establish standards in clock frequency, monitor location, and I/O conventions. The club formally adopted the 44 pin buss and the ACE buss pin assignment as the club standard, see Ipso Facto #14, and the use of a CPU clock rate of 1.758Mhz, derived from a standard colour burst crystal, and the use of a regulated buss for board design.

To facilitate other systems, each hardware board will have the provision for pin assignment changes by the use of two pads,with a removable strip between, on each buss lead at the edge connector. By removing the strip, users will be able to easily jumper the leads to any pin assignment desired. The club was directed to produce a 10 slot backplane, with buffering between slots 1 and 2 for data, address, timing and $\overline{MRD}$ and $\overline{MWR}$ leads to facilitate system expansion. A system interconnect from Netronics to ACE buss will also be given proiority. Two boards are already proto typed and debugged- an 8K Eprom board and a wire wrap board. A 16k 2114 RAM board with the MC6847 colour video circuit on board is currently being debugged. Boards in design include a Kansas City Digital Cassette interface, Real Time Clock, and dynamic RAM. Production and sale is expected to begin in the spring. All boards will be sold etched, drilled and plated through, with gold edge connectors. No parts will be sold with the boards.

On the subject of incorporation, the club directed the Executive to investigate further the insurance option, and to report the cost of insurance verse incorporation at the next meeting.

MINUTES OF MEETING - JAN. 8, 1980.

The president reported that liability insurance was not available at a reasonable price compared to the cost of incorporation. The Executive had examined the two options, and concluded that incorporation offer the best protection to the Executive, Editors, article Authors, and to the membership at large, should a law suit arise from slander, economic or physical damage to an individual using hardware or software writen up in Ipso Facto or from the use of one of the club's boards. The Executive will continue to develop the requirements for incorporation, and proceed with the process them selves to minimize costs.

A third Executive member is required for incorporation, and since both the President and Treasurer occupy two poistions, a third member was required. Bernie Murphy was nominated and elected unanimously to serve for the remainder of the club year.

Mike Franklin advised the members present of his intention to seek ammendment of the constitution at he annual meeting in May, to the effect that the executive be re-organized into 4 committees, an executive committee, comprising the President, past President, Secretary/Treasurer, and three members who would each represent one of the other committees- newsletter/publication, hardware/software projects, and club activities committees. Each committee would report to the Executive through their representative, and maintain their own budgets and minutes. Mike stressed the advantages of more autonomy and smaller woking units as the advantages of his proposal.

Fred Plethero advised the club of his intention to seek ammendment of the constitution to change the election of the executive for specific positions to election to the executive at large, with the executive choosing their own positions. Fred maintained that in this manner, people may be more willing to run for office with the prospect of being able to do what they liked, and possibly changing positions during the year to learn new jobs and make it more interesting.

Should any of the membership wish to seek ammendment of the constitution, they must advise the club of the subject of their ammendment, and have it distributed to the menbership prior to the May meeting.

Fred Feaver moved that the next meeting at Sheridan College begin at 7:30 (Feb 12) to permit more time for club discussion. The motion was approved.

The topic for the night was a demonstration by Vern Lowe of AMP of Canada, a manufacturer of electrical connectors. He presented a talk on the use of industrial connectors, and answered questions on particular applcations and on problems of buss expansion.

>

64K Decoding for Tektron Memory

Dan Carrigan

This circuit has been designed to decode and enable memory boards which only decode for 16K.

The decoder enables memory in 16K blocks. The two most significant bits of the address are latched by a latch (74LS74) and decoded by the AND gates (74LS00).

One immediate application that comes to mind is for people who own Tektron 7K memory boards. These boards only decode 16K of memory so if anyone wants to expand beyond 14K and still use the Tektron 7K memory, it can be done easily with this little circuit.
>

...Continued on Page: 54

...Continued from Page: 20

Unique Netronics Math Chip Function

DEG	- radians to degrees
INV	- inverse trig
IX	- I/X
EX	- E to X
LWX	- natural log
MS	- memory store
MR	- memory recall
PI	- PI constant
POP	- pop stack
ROLL	- roll stack
RAD	- degrees to Radians
SF 1 or 2	- set flag 1 or 2
SL or SR	- shift left or right - multiply or divide by 10
SQRT	- square root
TENX	- raise X to ten power, MAX 7
XM, XY	- math register date exchange
YX	- raise Y to X power

>

Cuddly Software: A Product Review

John McLaren
41 Walker Street
Coorparoo, Queensland,
Australia, 4151

I recently purchased CSOS and CSTP software from Cuddly Software in the United States. I was pleasantly surprised with the programs they work; they work well; they do precisely as the advertisement says they will; and the documentation is excellent. A full user manual and hex program listing are provided, with the manual giving good details of all the commands plus examples of usage. The guy (Paul) is obviously not making much money from the deal because the standard of documentation is high, with crisp reproduction the manual looks like it probably was printed using an offset printer with paper plates.

Oh! Yes, as a throw-away line. I happened to mention to him that I have two Elf II's: one without a VID board for the kids, the neighbourhood, and apparently the rest of Australia, and one with a VID board which is mine! mine! mine! So guess what happens. Cuddly Software send two versions of their programs. One for a non-VID system and one for a VID system, and all for the same price!! That is service.

The CSTP (Cuddly Software Trace Program) is good. Once you accept that 8-bit hardware has limitations, then you will find that CSTP is one of the best debug aids available for 1802's. As Paul says, it turns your machine into a "super 1802" - if the program being traced (manual or automatic single step by the way) wants input, then you have to give it input, if it wants an interrupt then you have to give it an interrupt. So, set it into automatic single step, set the time for each step, and sit back to watch your program go. Lovely!

The one which excites me, though, is CSOS (Cuddly Software Operating System) which is a rudimentary operating system (hereinafter denoted OS) that has great possibilities. Let's see it allows you to: add, change, delete and erase bytes in any page, and block-move data forward or backward within the entire address space(64K). It displays full and successive pages on your TV. It has 64x32, 64x64, and 64x128 resolution (if you have the memory) and allows you to run programs you enter or load behind it.

Additionally, there is a good discussion on how to alter (and by implication, design) the characters displayed on your screen if you don't happen to like them. Plus there are a few freebies in the form of patches which Cuddly Software will give you if you ask nicely.

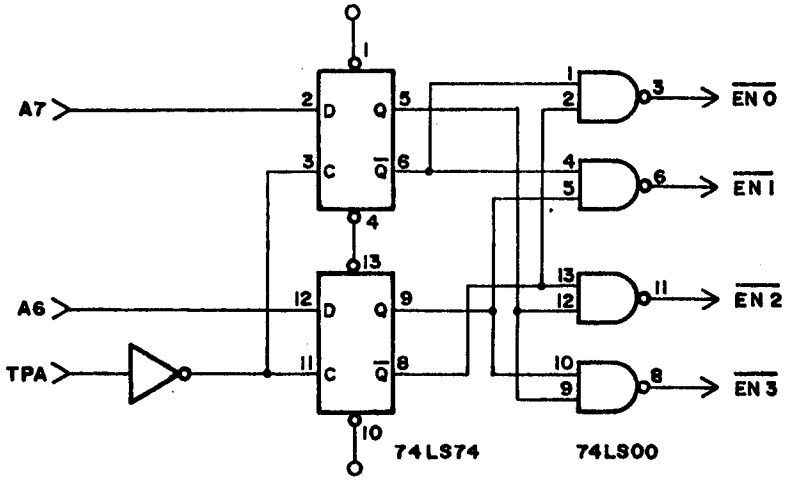
However, the beauty of OS is that it is task-driven through a task select table which can be altered (e.g. ERASE PAGE fills the page with 0 (use judiciously, please) but you could alter the function name to ZERO PAGE and relocate it within the task table). Thus, by implication, you could add your own routines (e.g. read/write totape or disk, write to printer, display graphics, jump to language, etc.) which means thatyou could come up with a pretty good operating system.

Finally, Paul is helpful. It you can't answer the questions on his configuration sheet, just tell him what you have got in the hardware line e.g. a standard Netronics Elf II with Giant Board, etc. He does the rest. All in all, a very good package and worth the money.

Editor's note:
We have had other favourable comments on Cuddly Software as well. Their address is: Cuddly Software
98 Thorndale Terrace
Rochester, N.Y., 14611

>

...Continued from Page: 51



16K BLOCK ENABLE DECODING

Errata: Day of the Week Program

Michael Mirsky VE3 INC
10 De Quincy Blvd.
Downsview, Ont. M3H 1Y5

Following are some corrections for the day of the week program that appeared in issue 15.

Line 170 was omitted. It should read:

```
170 IF A=999 GOTO 60
```

Line 80 should read:

```
80 K=(60+(10000/(100*M)))/100
```

Sorry about these errors, I don't know how I let them slip by. Also, it should be pointed out that the program works because of Tiny's integer arithmetic. If full Basic, with floating point arithmetic is used, the program will have to be modified using the INT function for every division in lines 80 to 120 so that all fractional parts are truncated as Tiny does in division.

>

PALO ALTO TINY BASIC INTERPRETER

Robert Edwards
104 Montreal Lane
Oak Ridge
TN, USA 37830

The Palo Alto Tiny Basic Interpreter for the 1802 is alive and well. Now that its checked out, the subsidy on its distribution is being withdrawn. Thus the current price for a copy of the source listing is $2.00 (US), which just covers expenses. I have distributed some 70 copies of the source code to date. I would like to hear from experimenters on their success (or lack of it) with the interpreter. Also, I would like to hear from US and Canadian owners of Netronics and Quest systems who could serve as distribution centers for cassette copies of the code. This means that experimenters can send the center a quality cassette and $1.00 (for postage and package) and would receive a copy of TBI1802 on the cassette in the return mail.

SUPER ISOLATOR--NEW PRODUCT RELEASE

ESP Electronic Specialists
171 South Main St.
Natick, Mass. USA
01760

IPSO FACTO has received the following new product announcement which may be of interest to readers:

"Severe AC power line spikes, surges, noises and hash are prevelent in many MicroProcessor installations. Operators are plagued with unexplained crashes, memory loss and program glitches. Disks, printer and processor often interact, aggravating the problem.

ELECTRONIC SPECIALISTS' recently announced SUPER ISOLATOR is designed to curb these severe electrical problems. Incorporating heavy duty surge/spike suppressors, the SUPER ISOLATOR features 3 individually dual-PI filtered 3-prong AC sockets. Equipment interactions are eliminated and disruptive/damaging line spikes and hash are controlled. The SUPER ISOLATOR can accomodate an 1875 watt load, with each socket capable of handling 1000 watts.

Severe power line spike/surge and hash control is combined with interaction-free operation for $79.95 (Model ISO-3)."

>

MEMBERSHIP RENEWAL/CHANGE OF ADDRESS

The dues for the new club year, September 1979 to August 1980 are as follows:

Canada $15.00
Elsewhere..... $18.00 (Canadian)

These new rates include provision for FIRST CLASS POSTAGE which should ensure that your copy of IPSO FACTO arrives as quickly as possible.

Do you have any friends? If so, they might well appreciate a gift subscription to IPSO FACTO.

New members are required to pay the full dues for the current year and will receive all the IPSO FACTO issues for the current year (September 1979 to August 1980). Back issues for IPSO FACTO 1-12 are completely sold out.

Please make cheque or postal money order payable in Canadian funds and send to:

ADDRESS CHANGE ☐
NEW SUBSCRIPTION ☐
RENEWAL ☐

"THE ASSOCIATION OF COMPUTER EXPERIMENTERS",
c/o M.E. Franklin, Treasurer,
A.C.E.,
24 Duby Road,
ACTON, Ontario, Canada.
L7J 2P1

FIRST NAME

LAST NAME

FIRST LINE OF ADDRESS

SECOND LINE OF ADDRESS

CITY & PROVINCE

POSTAL CODE

TELEPHONE NUMBER

ATTACH OLD MAILING LABEL
HERE FOR CHANGE OF ADDRESS